MICROSOFT™ MS-DOS™

Operating System 3.2

User's Reference

ULdos φ 62

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

Document No. 410630013-320-002-1285

Contents

Introduction

MS-DOS Version 3.2 Package Contents

1 disk with the following files:

ANSI.SYS
APPEND.EXE
ASSIGN.COM
ATTRIB.EXE
BACKUP.EXE
CHKDSK.EXE
COMMAND.COM
DEBUG.EXE
DISKCOMP.EXE
DISKCOPY.EXE
DRIVER.SYS
EDLIN.EXE
EXE2BIN.EXE
FC.EXE
FDISK.EXE
FIND.EXE
FORMAT.EXE
GRAFTABL.EXE
GRAPHICS.COM
IO.SYS (hidden file)
JOIN.EXE
LABEL.EXE
LINK.EXE
MODE.COM
MORE.COM
MSDOS.SYS (hidden file)
PRINT.EXE
RAMDRIVE.SYS
RECOVER.EXE
REPLACE.EXE
RESTORE.EXE
SHARE.EXE
SORT.EXE
SUBST.EXE
SYS.COM
TREE.EXE
XCOPY.EXE

3 manuals:

The MS-DOS User's Guide
The MS-DOS User's Reference Manual
The MS-DOS Programmer's Reference Manual

INTRODUCTION

1.0 WHAT IS MS-DOS?

Microsoft(R) MS-DOS(R) is a disk operating system for the 8086 family of computers. Through MS-DOS, you communicate with the computer, disk drives, and printer, managing these resources to your advantage.

2.0 ABOUT THIS MANUAL

This manual is a reference manual for MS-DOS. Two companion manuals tell you more about MS-DOS. These manuals are the _MS-DOS User's Guide_ and the _MS-DOS Programmer's Reference Manual_. We assume that you have read the _MS-DOS User's Guide_ and know how to start MS-DOS, how to copy, delete, and rename files, how to make copies of disks, and how to run applications. You should have already made a backup copy of your MS-DOS disk and stored the original in a safe place. Refer to the _MS-DOS User's Guide_ to learn how to make a backup copy of your disk.

The following table will help you to use this manual:

If you want to know...	Turn to...
About naming files	Chapter 1
About multilevel directories	Chapter 1
About paths	Chapter 1
What a command is	Chapter 2
How to make a batch file	Chapter 2
What a command does	Chapter 3
About editing keys	Chapter 4
How to use EDLIN, the line editor	Chapter 5
How to use FC, the file comparison utility	Chapter 6
How to use LINK, a link utility	Chapter 7
How to use DEBUG, a debugging utility	Chapter 8
What to do if you have one drive	Appendix A
What a disk error means	Appendix B
About ANSI escape sequences	Appendix C
What a CONFIG.SYS file does	Appendix D
What an error message means	Appendix E
How to configure a hard disk	Appendix F
About installable device drivers	Appendix G

3.0 MS-DOS FILES

The MS-DOS disk contains the following files:

Filename	Function of File
COMMAND.COM	MS-DOS command processor
* MSDOS.SYS	MS-DOS operating system
* IO.SYS	Hardware-operating system interface
ANSI.SYS	Sets ANSI escape sequences
APPEND.EXE	Sets a search path for data files
ASSIGN.COM	Assigns a drive letter to a different drive
ATTRIB.EXE	Sets file attributes
BACKUP.EXE	Backs up files
CHKDSK.EXE	Checks disks
CONFIG.SYS	Configures system
DEBUG.EXE	Debugger
DISKCOMP.EXE	Compares disks
DISKCOPY.EXE	Copies disks
DRIVER.SYS	Configures installable block device drivers
EDLIN.EXE	Line editor
EXE2BIN.EXE	Converts .EXE files (optional file)
FC.EXE	Compares files
FDISK.EXE	Configures hard disks for MS-DOS
FIND.EXE	Finds a string in a list of files or standard input
FORMAT.EXE	Formats disks
GRAFTABL.COM	Loads a table of graphics characters
GRAPHICS.COM	Prepares for printing graphics
JOIN.EXE	Joins a disk drive to a specific pathname
LABEL.EXE	Labels a disk
LINK.EXE	Linker
MODE.COM	Sets output and display modes
MORE.COM	Reviews text
PRINT.EXE	Prints files
RAMDRIVE.SYS	Installs a virtual disk
RECOVER.EXE	Recovers disks
REDIR.EXE	Accesses files
REPLACE.EXE	Replaces previous version of files
RESTORE.EXE	Restores files
SHARE.EXE	Shares files
SORT.EXE	Sorts text
SUBST.EXE	Creates a string alias
SYS.COM	Transfers system
TREE.EXE	Lists files and subdirectories
XCOPY.EXE	Copies files and subdirectories

You will recognize this list of files when you issue a Dir (Show Directory) command, described in the MS-DOS User's Guide. The two files preceded by an asterisk (*) are "hidden" files and do not appear when you enter a Dir command.

4.0 WHAT'S NEXT?

In the next chapter, you will learn about files and directories.

Chapter 1
Files and Directories

CHAPTER 1

FILES AND DIRECTORIES

Note

Before you read this chapter, you should already know how to start MS-DOS, format and make backup copies of disks, copy and delete files, and run programs. If you are unfamiliar with how to do any of these, please refer to the MS-DOS User's Guide for information.

1.1 FILES

1.1.1 What is a File?

A file is a collection of related information. A file on your disk can be compared to a file folder in a desk drawer. For example, file folders might contain business letters from clients and important memos from associates.

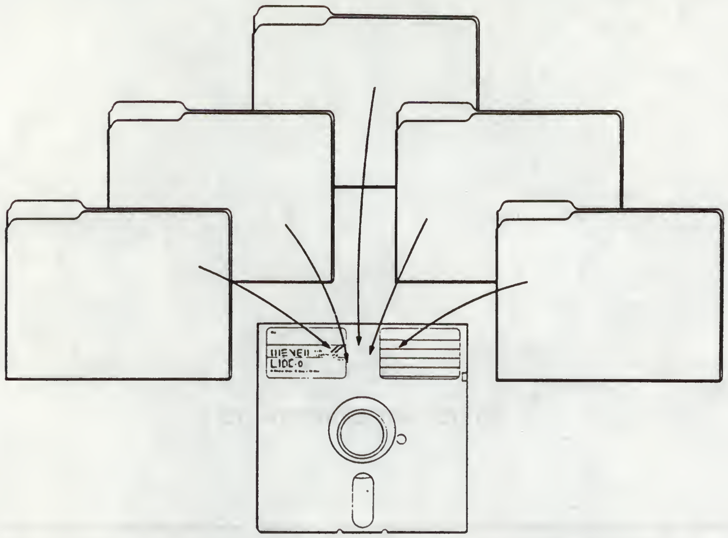


Figure 1.1. Disk Files are Like Paper Files

Files on your disks could also contain business letters and memos.

All programs, text, and data on your disk reside in files and each file has a unique name. You refer to files by their filenames. In this chapter, you will learn how to name your files.

You create a file each time you enter and save data or text at your terminal. Files are also created when you write programs and save them on your disks.

1.1.2 How MS-DOS Keeps Track of Your Files

The names of files are kept in directories on a disk. These directories also contain information on the size of the files and the dates that they were created and updated. The directory you are working in is called your working directory.

An additional system area, called the File Allocation Table, keeps track of the location of your files on the disk. It also allocates the free space on your disks, so that you can create new files.

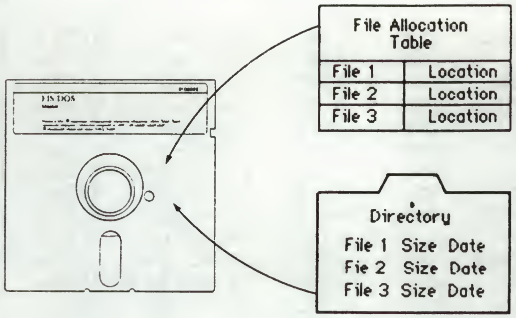


Figure 1.2. Directory and File Allocation Table

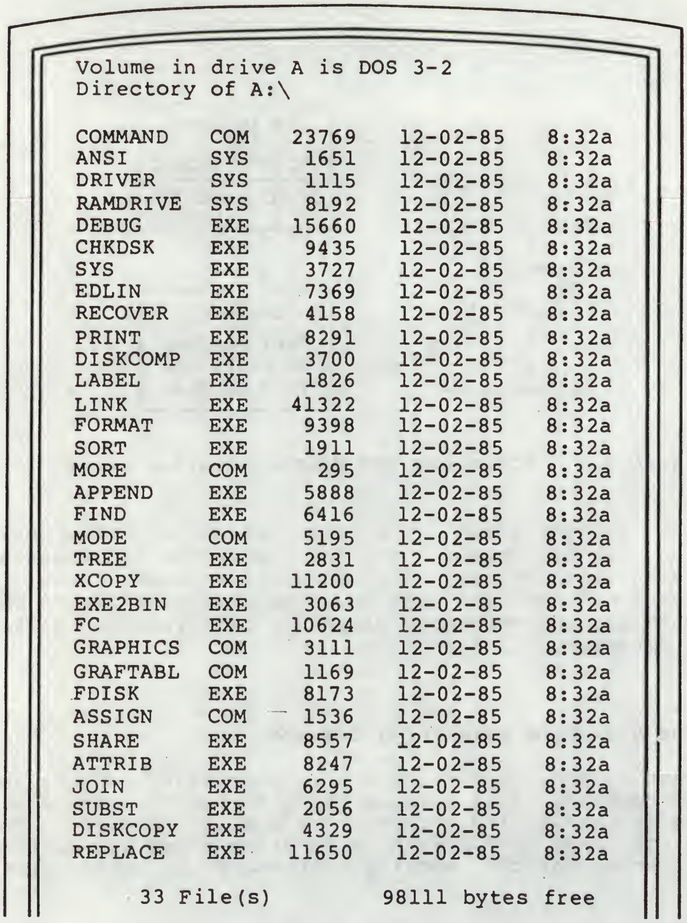
These two system areas, the directories and the File Allocation Table, enable MS-DOS to recognize and organize the files on your disks. The File Allocation Table is copied onto a new disk when you format it with the MS-DOS Format command, and one empty directory is created, called the root directory.

1.1.3 The Dir (Show Directory) Command

If you want to know what files are on your disk, you can use the Dir command. This command tells MS-DOS to display all the files in the working directory on a specific disk. For example, if your MS-DOS disk is in drive A and you want to see the listing for the working directory on that disk, type:

```
        dir a:
        ─┬─ ┌─
         │  └──────"of the disk in
    This says       drive A."
    "show me
    the directory"
```

MS-DOS responds with a directory listing of all the files in the working directory on your MS-DOS disk. The display should look similar to this:



```
Volume in drive A is DOS 3-2
Directory of A:\

COMMAND   COM   23769   12-02-85   8:32a
ANSI      SYS    1651   12-02-85   8:32a
DRIVER    SYS    1115   12-02-85   8:32a
RAMDRIVE  SYS    8192   12-02-85   8:32a
DEBUG     EXE   15660   12-02-85   8:32a
CHKDSK    EXE    9435   12-02-85   8:32a
SYS       EXE    3727   12-02-85   8:32a
EDLIN     EXE    7369   12-02-85   8:32a
RECOVER   EXE    4158   12-02-85   8:32a
PRINT     EXE    8291   12-02-85   8:32a
DISKCOMP  EXE    3700   12-02-85   8:32a
LABEL     EXE    1826   12-02-85   8:32a
LINK      EXE   41322   12-02-85   8:32a
FORMAT    EXE    9398   12-02-85   8:32a
SORT      EXE    1911   12-02-85   8:32a
MORE      COM     295   12-02-85   8:32a
APPEND    EXE    5888   12-02-85   8:32a
FIND      EXE    6416   12-02-85   8:32a
MODE      COM    5195   12-02-85   8:32a
TREE      EXE    2831   12-02-85   8:32a
XCOPY     EXE   11200   12-02-85   8:32a
EXE2BIN   EXE    3063   12-02-85   8:32a
FC        EXE   10624   12-02-85   8:32a
GRAPHICS  COM    3111   12-02-85   8:32a
GRAFTABL  COM    1169   12-02-85   8:32a
FDISK     EXE    8173   12-02-85   8:32a
ASSIGN    COM    1536   12-02-85   8:32a
SHARE     EXE    8557   12-02-85   8:32a
ATTRIB    EXE    8247   12-02-85   8:32a
JOIN      EXE    6295   12-02-85   8:32a
SUBST     EXE    2056   12-02-85   8:32a
DISKCOPY  EXE    4329   12-02-85   8:32a
REPLACE   EXE   11650   12-02-85   8:32a

     33 File(s)        98111 bytes free
```

Note

Two MS-DOS system files, IO.SYS and MSDOS.SYS, are "hidden" files and will not appear when you issue the Dir command.

You can also get information about any file on your disk by typing Dir and a filename. For example, if you have created a file named MYFILE.TXT, the command:

```
dir myfile.txt
```

gives you a display of all the directory information (name of file, size of file, date created or edited) for the file MYFILE.TXT.

For more information on the Dir command, refer to Chapter 3, "MS-DOS Commands."

1.1.4 The Chkdsk (Check Disk) Command

The MS-DOS Chkdsk command is used to check your disks for consistency and errors, much like a secretary proofreading a letter. Chkdsk analyzes the directories and the File Allocation Table on the disk that you specify. It then produces a status report of any problems, such as files which have a non-zero size in the directory but really have no data in them.

To check the disk in drive A, type:

```
chkdsk a:
```

MS-DOS displays a status report and any errors that it has found. An example of this display and more information on Chkdsk can be found in the description of the Chkdsk command in Chapter 3. You should run Chkdsk occasionally for each disk to make sure the files on your disk are OK.

1.2 HOW TO NAME YOUR FILES

The name of a typical MS-DOS file looks like this:

$$\underbrace{\texttt{newfile}}_{\text{filename}}\texttt{.}\underbrace{\texttt{doc}}_{\text{filename extension}}$$

The name of a file consists of two parts. The <u>filename</u> is NEWFILE and the <u>filename</u> <u>extension</u> is .DOC.

A filename can be from 1 to 8 characters long. The filename extension is optional, and can be three or fewer characters. It must be separated from the filename by a period. You can type any filename in small or capital letters. MS-DOS then translates these letters into uppercase characters. Some examples of filenames are:

```
ACCOUNTS.FEB
BUDGET.84
SMITHCO.LTR
CHAPTER1.NVL
SCHEDULE
```

In addition to typing the filename and the filename extension, you may need to include a drive name. A drive name tells MS-DOS to look on the disk in a specific drive to find the filename typed. For example, to find directory information about the file NEWFILE.DOC, located on the disk in drive B (when drive B is NOT the current [default] drive), type the following command:

```
dir b:newfile.doc
```

Directory information (name, size, date, and time created) about the file NEWFILE.DOC are displayed on your screen.

If drive A is the default drive, MS-DOS automatically searches the disk in drive A for the filename NEWFILE; so it is not necessary to type the drive name. A drive name is needed if you want to tell MS-DOS to look on a different drive to find a file.

Your filenames will probably be made up of letters and numbers, but other characters are allowed, too. Valid characters for filename extensions are the same as those for filenames. Here is a complete list of the letters and symbols you can use in filenames and extensions:

```
A-Z a-z  0-9   $   &

%    '    (    )   -   @

^    {    }   ~   `  !   #
```

1.3 INVALID FILENAMES

MS-DOS treats some device names specially, and certain three-letter names are reserved for the names of these devices. These three-letter names cannot be used as filenames, but they can be used as extensions. You must not name your files any of the following:

AUX	Used when referring to input from, or output to, an auxiliary device (such as a printer or a disk drive).
CON	Used when referring to keyboard input, or to output to the terminal console (screen).
PRN	Used when referring to the printer device.
NUL	Used when you do not want to create a particular file, but the command requires an input or output filename.

Even if you add device designations or filename extensions to these filenames, MS-DOS still assumes that they refer to the devices listed above. For example, CON.XXX still refers to the console and cannot be used as the name of a disk file.

1.4 WILDCARDS

Two special characters (called wildcards) can be used when you are searching for files on a disk: the asterisk (*) and the question mark (?). These special characters give you greater flexibility when using filenames in MS-DOS commands.

1.4.1 The ? Wildcard

A question mark (?) in a filename or filename extension means that any character can occupy that position. For example, the MS-DOS command:

```
     dir test?run.exe
```

lists all directory entries on the default drive that have eight characters in the filename, begin with TEST, have any character in the next position, end with the letters RUN, and have a filename extension of .EXE. Here are some examples of files that might be listed by the above Dir command:

```
     TEST1RUN.EXE
     TEST2RUN.EXE
     TESTBRUN.EXE
```

1.4.2 The * Wildcard

An asterisk (*) in a filename or filename extension means that any character can occupy that position <u>or any of the remaining positions in the filename or extension</u>. For example,

```
dir test*.exe
```

will list all directory entries on the default drive with filenames that begin with the characters TEST and have an extension of .EXE. Here are some examples of files that might be listed by the above Dir command:

```
TEST1.EXE
TEST2RUN.EXE
TEST6RUN.EXE
TESTALL.EXE
```

Important

The wildcard designation *.* refers to all files on the disk. Note that this can be very powerful and destructive when used in MS-DOS commands. For example, the command:

```
del *.*
```

deletes all files on the default drive, regardless of filename or extension.

Examples:

To list the directory entries for all files named ACCOUNTS on drive A (regardless of their filename extensions), simply type:

```
dir a:accounts.*
```

To list the directory entries for all files with filename extensions of .TXT (regardless of their filenames) on the disk in drive B, type:

```
dir b:*.txt
```

This command is useful if, for example, you have given all your text files a filename extension of .TXT. By using the Dir command with wildcard characters, you can get a listing of all your text files even if you do not remember all of their filenames.

1.5 HOW TO PROTECT YOUR FILES

MS-DOS is a powerful and useful tool when you process personal and business information. As with any computer, errors can occur and information may be misused. If you are doing work that cannot be replaced or requires a high level of security, you should take steps to ensure that your programs are protected from others using, modifying, or even deleting them. Simple measures you can take--such as removing your disks when they are not in use, keeping backup copies of valuable information, and installing your equipment in a secure facility--can help you keep your files secure.

1.6 DIRECTORIES

The names of your files are kept in a directory on each disk. The directory also contains information on the size of the files, and the dates they were created and updated.

When there is more than one user on your computer, or when you are working on several different projects, the number of files in the directory can become large and unwieldy. You may want your own files kept separate from a co-worker's, or you may want to organize your programs into categories that are convenient for you.

In an office, you can separate files by putting them in different filing cabinets; in effect, creating different directories of information. MS-DOS lets you organize the files on your disks into directories. Directories are a way of dividing your files into convenient <u>groups</u> of files. For example, you may want all of your accounting programs in one directory and text files in another. Any one directory can contain any reasonable number of files, and it may also contain other directories (referred to as subdirectories). This method of organizing your files is called a multilevel directory structure.

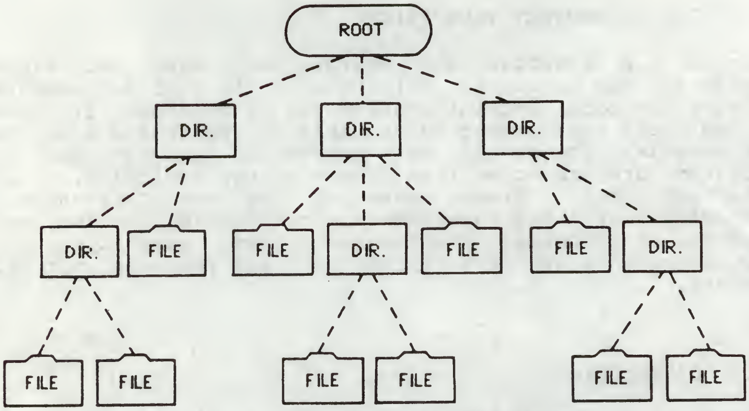


Figure 1.3. Multilevel Directory Structure

A multilevel directory structure can be thought of as a "tree" structure: directories are branches of the tree and files are the leaves, except that the "tree" grows downward; that is, the "root" is at the top. The root is the first level in the directory structure. It is the directory that is automatically created when you format a disk and start putting files on it. You can create more directories and subdirectories by following the instructions later in this chapter.

The directory structure grows as you create new directories for groups of files, or for other people on the system. Within each new directory, files can be added, or new subdirectories can be created.

It is possible for you to "travel" around this structure to find any file in the system by starting at the root, then traveling down any of the branches to the desired file. Conversely, you can start where you are within the file system and travel toward the root.

The directory you are in at any time is called the "working directory." The filenames discussed earlier in this chapter are relative to your working directory, and do not apply to any other directories in the structure. Thus, when you turn on your computer, you are "in" the working directory. Unless you take special action when you create a file, the new file is created in the working directory. Users can have files of the same name that are unrelated because each is in a different directory. See Figure 1.4, which illustrates a typical multilevel directory structure.

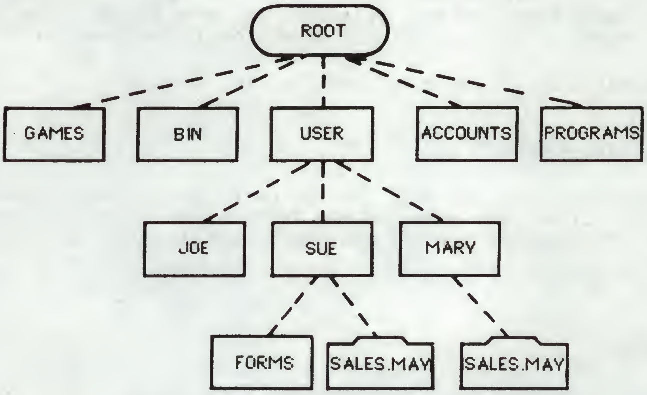


Figure 1.4. A Sample Multilevel Directory Structure

The root directory is the first level in the directory structure. You can create subdirectories from the root by using the Mkdir command (refer to Chapter 3, "MS-DOS Commands," for information on Mkdir). In this example, five subdirectories of ROOT have been created. These include:

1. A directory of games, named GAMES.

2. A directory of all external commands, named BIN.

3. A USER directory containing separate subdirectories for all users of the system.

4. A directory containing accounting information, named ACCOUNTS.

5. A directory of programs, named PROGRAMS.

Suppose three co-workers, Joe, Sue, and Mary, each have their own directories, which are subdirectories of the USER directory. Sue has a subdirectory named FORMS. Sue and Mary have files in their directories, each named SALES.MAY. Mary's SALES.MAY file is unrelated to Sue's.

This organization of files and directories is not important if you only work with files in your own directory, but if you work with someone else, or on several projects at once, the multilevel directory structure becomes very handy. For example, you could get a list of the files in Sue's FORMS directory by typing:

```
     dir \user\sue\forms
```

Note that a backslash mark (\) is used to separate directories from other directories and files. The first slash indicates the separation from the root directory.

To find out what files Mary has in her directory, you could type:

```
    dir \user\mary
```

This command tells MS-DOS to travel down the directory structure from the root to the USER directory, and then to the MARY directory, and to display all filenames in the MARY directory.

1.7 PATHS

When you use multilevel directories, you must tell MS-DOS where the files are located in the directory structure. Both Mary and Sue, for example, have files named SALES.MAY. Each will have to tell MS-DOS in which directory her file resides when she wants to use it. They can do this by giving MS-DOS a pathname to the file.

1.7.1 Pathnames

A pathname is a sequence of directory names followed by a filename, each separated from the previous one by a backslash (\).

The general format of pathnames is:

```
     [<directoryname>]\[<directoryname>...]\[<filename>]
```

A pathname may contain any number of directory names. If a pathname begins with a slash, MS-DOS searches for the file beginning at the root (or top) of the directory structure. Otherwise, MS-DOS begins at the working (current) directory and searches downward from there.

Note

Some MS-DOS commands, such as Dir and Print, cannot accept pathnames with more than 63 characters, even though you can create directories that are longer than this limit. You must change directories to access these files.

The pathname of Sue's SALES.MAY file is:

```
\USER\SUE\SALES.MAY
```

The pathname of Mary's SALES.MAY file is:

```
\USER\MARY\SALES.MAY
```

When you are in your working directory, a filename and its corresponding pathname may be used interchangeably. Some sample names are:

\	Indicates the root directory.
\PROGRAMS	Directory under the root directory that contains program files.
\USER\MARY\FORMS\1A	A typical full pathname. This one is file named 1A in the directory named FORMS belonging to Mary.
SALES.MAY	Name of a file in the working directory.

Each directory that has subdirectories beneath it is called a "parent directory." MS-DOS provides special shorthand notations for the working directory and the parent directory (one level up) of the working directory:

	MS-DOS uses this "shorthand" name to indicate the name of the working directory in all multilevel directory listings. MS-DOS automatically creates this entry when a directory is made.
..	The shorthand name of the working directory's parent directory (one level up). If you type:

```
dir ..
```

MS-DOS lists the files in the parent directory of your working directory.

If you type:

```
dir ..\..
```

MS-DOS lists the files in the parent's PARENT directory.

1.7.2 Paths and External Commands

External commands reside on disks as program files. They must be read from the disk before they can execute. (For more information on external commands, refer to Chapter 2, "Learning About Commands.")

When you are working with more than one directory, it is convenient to put all MS-DOS external commands into a separate directory so they do not clutter your other directories. When you issue an external command to MS-DOS, MS-DOS immediately checks your working directory to find that command. You must tell MS-DOS in which directory these external commands reside. This is done with the Path command.

For example, if you are in a working directory named \BIN\PROG, and all MS-DOS external commands are in \BIN, you must tell MS-DOS to choose the \BIN path to find the Format command. The command:

```
path \bin
```

tells MS-DOS to search in your working directory <u>and</u> the \BIN directory for all commands. You only have to specify this path once to MS-DOS during each computer session. If you want to know what the working path is, type the word:

```
path
```

to display the working path on the screen.

You can establish a search path by including the Path Command in a special file named AUTOEXEC.BAT. Refer to Chapter 2, "Learning About Commands," for more information on the AUTOEXEC.BAT file.

For more information on the MS-DOS Path command, refer to Chapter 3, "MS-DOS Commands."

1.7.3 Paths and Internal Commands

Internal commands are the simplest, most commonly used commands. They execute immediately because they are loaded into your computer's memory when you start MS-DOS. (For more information on internal commands, refer to Chapter 2, "Learning About Commands.")

Some internal commands can use paths. The commands, Copy, Dir, Del, and Type, have greater flexibility when you specify a pathname after the command.

The format of these commands is shown below.

Copy <pathname> <pathname>
If the second pathname is a directory, all files are copied into that directory.

Example:

```
copy \bin\user\joe  \accounts
```

Del <pathname>
If the pathname is a directory, all the files in that directory are deleted. The prompt "Are you sure (Y/N)?" is displayed if you try to delete a path. Type Y (for Yes) to complete the command, or type N (for No) to stop the command.

Example:

```
del \user\joe
```

Dir <pathname>
Displays the directory for a specific path.

Example:

```
dir \user\joe
```

Type <pathname>
You·must specify a filename in a path for this command. MS-DOS displays the file on your screen in response to the Type command.

Example:

```
type \user\sue\report.doc
```

1.8 USING DIRECTORIES

The following sections describe how to display, change, and delete your working directory. You will also learn how to create directories and subdirectories.

1.8.1 How to Display Your Working Directory

All commands are executed while you are in your working directory. You can find out the name of the directory you are in by issuing the MS-DOS command Chdir (Change Directory) with no pathname. For example, if your working directory is \USER\JOE, when you type:

```
chdir
```

and press the Return key, you would see:

```
A:\USER\JOE
```

This is your working drive plus the working directory (\USER\JOE).

Shortcut

You can use the letters "cd" for the "chdir" command to save time. The command "cd user\joe" is the same as "chdir \user\joe".

If you now want to see what is in the \USER\JOE directory, you can use the MS-DOS Dir command. The following is an example of the display you might receive from the Dir command for a subdirectory:

```
Volume in drive A has no ID
Directory of A:\USER\JOE

.                   <Dir>        12-02-85    10:09a
..                  <Dir>        12-02-85    10:09a
TEXT                <Dir>        12-02-85    10:09a
FILE1    COM           5243      12-02-85    11:30a
        4 File(s)    836320 bytes free
```

A volume ID for this disk was not assigned when the disk was formatted. (For information on assigning a volume ID to a disk, turn to the Format command in Chapter 3, "MS-DOS Commands.") Note that MS-DOS lists both files and directories in this output. As you can see, Joe has a subdirectory named TEXT. The "." means the working directory \USER\JOE, and the ".." is short for the parent directory \USER. FILE1.COM is a file in the \USER\JOE directory. All of these directories and files are on the disk in drive A.

Note

Because MS-DOS considers a directory to be just a special type of file, you cannot give a subdirectory the same name as a file in that directory. For example, if you have a path \BIN\USER\JOE where JOE is a subdirectory, you cannot create a file named JOE in the \BIN\USER directory

1.8.2 How to Create a Directory

To create a subdirectory in your working directory, use the Mkdir (Make Directory) command. For example, to create a new directory named NEWDIR under your working directory, simply type:

```
     mkdir newdir
```

After MS-DOS runs this command, a new directory exists under your working directory. You can also create directories anywhere in the directory structure by specifying Mkdir and then the path of the directory you want to create. MS-DOS automatically creates the "." and ".." entries in the new directory.

There is a limit on the length of a pathname:

```
                  a:\attorney\jacobson\contracts\ellis.new

     from here<------------------------------------>to here
```

This pathname cannot be over 63 characters long.

To put files in the new directory, use the MS-DOS line editor, EDLIN. Chapter 5, "The Line Editor (EDLIN)," describes how to use EDLIN to create and save files. You can also create and save files if you have purchased a word processing program such as Microsoft(R) Word.

1.8.3 How to Change Your Working Directory

Changing from your working directory to another directory is very easy in MS-DOS. Simply type the Chdir (Change Directory) command and then a pathname. For example, type:

```
chdir \user
```

Then press the Return key. MS-DOS changes the working directory to \USER. You can specify any pathname after the command to "travel" around the directory structure. The command:

```
chdir ..
```

always puts you in the parent directory of your working directory.

1.8.4 How to Delete a Directory

To delete a directory in the structure, use the MS-DOS Rmdir (Remove Directory) command. For example, to remove the NEWDIR directory from the working directory, type:

```
rmdir newdir
```

The NEWDIR directory must be <u>empty</u> except for the "." and ".." entries before it can be removed; this prevents you from accidentally deleting files and directories. You can delete any directory by specifying its pathname. To remove the \USER\JOE directory, make sure that it has only the "." and ".." entries, then type:

```
rmdir \user\joe
```

To remove all the <u>files</u> in a directory (except for the . and .. entries), type Del and then the pathname of the directory. For example, to delete all files in the \USER\SUE directory, type:

```
del \user\sue
```

MS-DOS prompts:

```
Are you sure (Y/N)?
```

If you are sure you want to delete all the files in the directory, type Y (for Yes). Type N (for No) to stop the command.

You cannot delete the "." and ".." entries. They are created by MS-DOS as part of the multilevel directory structure.

1.9 HOW TO RENAME A DIRECTORY

There is no command to rename a directory in MS-DOS. You can, however, rename a directory that has no subdirectories by following these steps:

1. Create a new directory with the new name using the Mkdir command.

2. Copy all files from the old directory to the new directory with the Copy *.* command.

3. Delete the contents of the old directory with the Del *.* command.

4. Delete the old directory with the Rmdir command.

Example:

To rename the \user\joe directory \user\sue:

1. Type:

   ```
   mkdir \user\sue
   ```

   and press the Return key.

2. Type:

   ```
   copy \user\joe\*.* \user\sue
   ```

3. Type:

   ```
   del \user\joe\*.*
   ```

   (Type Y in response to the prompt "Are you sure?")

4. Type:

   ```
   rmdir \user\joe
   ```

1.10 SUMMARY OF COMMANDS IN THIS CHAPTER

COMMAND	PURPOSE	EXAMPLE
Dir	Displays a directory	dir a:
Chkdsk	Checks disks	chkdsk b:
Path	Sets MS-DOS search path	path \deb\games
Chdir	Displays working directory; changes directories	chdir \user\bill
Mkdir	Makes a new directory	mkdir \user\sally
Rmdir	Removes a directory	rmdir \bin\games

1.11 WHAT'S NEXT?

In the next chapter, you will learn about MS-DOS commands.

Chapter 2
Learning About Commands

CHAPTER 2

LEARNING ABOUT COMMANDS

2.1 WHAT IS A COMMAND?

A command is a way of communicating with the computer. By typing MS-DOS commands at your terminal, you can ask the computer to perform useful tasks. There are MS-DOS commands that:

Compare, copy, display, delete, and rename files
Copy and format disks
Execute programs
Analyze and list directories
Enter date, time, and remarks
Set various printer and screen options
Copy MS-DOS system files to another disk
Request MS-DOS to wait for a specific period of time

2.2 TYPES OF MS-DOS COMMANDS

There are two types of MS-DOS commands:

Internal commands
External commands

Internal commands are the simplest, most commonly used commands. You cannot see these commands when you do a directory listing on your MS-DOS disk; they are part of a large file named COMMAND.COM. When you type internal commands, they execute immediately. The following internal commands are described in Chapter 3, "MS-DOS Commands":

Break	Del	Mkdir	Set
Chdir	Dir	Path	Shift
Cls	Echo	Pause	Time
Copy	Exit	Prompt	Type
Ctty	For	Rem	Ver
Date	Goto	Ren	Verify
	If	Rmdir	Vol

External commands are on disks as program files. They must be read from a disk before they can execute.

Any filename with a filename extension of .COM, .EXE or .BAT is considered an external command. For example, programs such as FORMAT.COM and DISKCOPY.COM are external commands. Because all external commands are files, you can create commands and add them to MS-DOS. Programs that you create with most languages (including assembly language) are .EXE (executable) files.

When you type an external command, do not include its filename extension. The following external commands are described in Chapter 3, "MS-DOS Commands":

Append	Fdisk	Print
Assign	Find	Recover
Attrib	Format	Restore
Backup	Graftabl	Share
Chkdsk	Graphics	Sort
Command	Join	Subst
Diskcomp	Label	Sys
Diskcopy	Mode	Tree
Exe2bin	More	

2.3 COMMAND OPTIONS

You can use options in your commands to give MS-DOS extra information. If you do not include some options, MS-DOS provides a value called a "default value." Refer to individual command descriptions in Chapter 3 for the default values.

Square brackets ([ ]) mean that an item is optional. Angle brackets (< >) mean that you supply the text for the item. Note that the drive name is not required unless you need to indicate to MS-DOS which disk to search for a specific file.

The following is the format of all MS-DOS commands:

```
Command [options...]
```

where [options...] can be one of the following:

drive:	Refers to disk drive name.
filename	Refers to any name for a disk file, including the filename extension if there is one. The filename option does <u>not</u> refer to a device or to a disk drive name.
pathname	Refers to a pathname or filename in the following general format: `[<directory>]\[<directory>...]\[<filename>]`
switches	Switches are options that control MS-DOS commands. They are preceded by a forward slash (for example, /p).
arguments	Provide more information to MS-DOS commands. You usually choose between arguments, for example, ON or OFF.

2.4 THINGS YOU SHOULD KNOW ABOUT COMMANDS

The following information applies to all MS-DOS commands:

1. Commands are usually followed by one or more options, such as a filename.

2. Commands and options may be typed in uppercase or lowercase letters, or any combination of the two.

3. Commands and options must be separated by certain characters or spaces. Because they are easiest, you usually use the space and comma to separate commands from options. For example:

   ```
   del newfile.old newfile.txt
   rename,thisfile thatfile
   ```

   You can also use the semicolon (;), the equal sign (=), or the Tab key between MS-DOS commands and their options.

   In this manual, we will use a space between commands and options, such as:

   ```
   copy memol memo2
   ```

4. Commands take effect only after you have pressed the Return key.

5. When instructions say "Press any key," you can press any alpha (A-Z) or numeric (0-9) key, or the Spacebar.

6. You must include the filename extension when referring to a file that has a filename extension.

7. You can stop commands while they are running by pressing Control-C.

8. When commands produce a large amount of output on the screen, the display automatically scrolls to the next screen. You can press Control-S to suspend the scrolling. Press Control-S again to view the rest of the display on the screen.

9. MS-DOS editing and function keys can be used when typing commands. Refer to Chapter 4, "MS-DOS Editing and Function Keys," for a complete description of these keys.

10. Disk drives are referred to as source drives and destination drives. A source drive is the drive you will be transferring information from. A destination drive is the drive you will be transferring information to.

2.5 INPUT AND OUTPUT

MS-DOS always assumes that input comes from the keyboard and output goes to the screen. However, the flow of command input and output can be redirected. Input can come from a file rather than a keyboard, and output can go to a file or to a line printer instead of to the screen. Also, "pipes" can be created that allow output from one command to become the input to another. Redirection and pipes are discussed in the next sections.

2.5.1 How to Redirect Your Output

Most commands produce output that is sent to your terminal screen. You can send this information to a file by using a greater-than sign (>) in your command. For example, the command:

```
dir
```

displays a directory listing of the disk in the default drive on the screen. The same command can send this output to a file named NEWFILES by typing the output file on the command line following a greater-than sign (>):

```
dir >newfiles
```

If the file NEWFILES doesn't exist, MS-DOS creates it and stores your directory listing in it. If NEWFILES already exists, MS-DOS overwrites what is in the file with the new data.

If you want to append your directory or a file to another file (instead of replacing the entire file), two greater-than signs (>>) can be used to tell MS-DOS to append the output of the command (such as a directory listing) to the end of a specified file. The command:

```
dir >>newfiles
```

appends your directory listing to an existing file named NEWFILES. If NEWFILES doesn't exist, it is created.

It is often useful to have input for a command come from a file rather than from a terminal. This is possible in MS-DOS by using a less-than sign (<) in your command. For example, the command:

```
sort <names >list1
```

sorts the file NAMES and sends the sorted output to a file named LIST1. (For more information on the Sort command, see Chapter 3, "MS-DOS Commands.")

2.5.2 Filters

A filter is a command that reads your input, transforms it in some way, and then outputs it, usually to your screen or to a file. In this way, the data is said to have been "filtered" by the program. Since filters can be put together in many different ways, a few filters can take the place of many specific commands.

MS-DOS filters include Find, More, and Sort. Their functions are described below:

Find	Searches for some text in a file.
More	Displays text one screenful at a time.
Sort	Sorts text.

You can see how these filters are used in the next section.

2.5.3 Command Piping

If you want to give more than one command at a time to the system, you can "pipe" commands to MS-DOS. For example, sometimes you may want the output of one program sent as the input to another program. A typical case would be a program that produces output in columns. You might want to have these columns sorted.

Piping is done by separating commands with the pipe symbol, which is the vertical bar (|). For example, the command:

```
dir | sort
```

displays an alphabetically sorted listing of your directory on the screen. The vertical bar causes all output generated by a command on the left side of the bar to be sent to the right side of the bar for processing.

You can also use piping when you want to send output to a file. If you want your directory sorted and sent to a new file (for example, DIREC.FIL), you could type:

```
dir | sort >direc.fil
```

MS-DOS creates a file named DIREC.FIL on your default drive. DIREC.FIL contains a sorted listing of the directory on the default drive, since no other drive was specified in the command. To specify a drive other than the default drive (for example, drive B), type:

```
     dir | sort >b:direc.fil
```

This sends the sorted data to a file named DIREC.FIL on drive B.

A "pipeline" may consist of more than two commands. For example:

```
     dir | sort | more
```

sorts your directory, shows it to you one screen at a time, and puts --MORE-- at the bottom of your screen when there is more output to be seen.

You will find many uses for piping commands and filters.

2.6 BATCH PROCESSING

Often you may find yourself typing the same sequence of commands over and over to perform some common task. With MS-DOS, you can put the command sequence into a special file called a batch file, and execute all the commands simply by typing the name of the batch file. "Batches" of your commands in such files are processed as if they were typed from the keyboard. Each batch file must be named with the .BAT extension, and is executed by typing the filename <u>without its extension</u>.

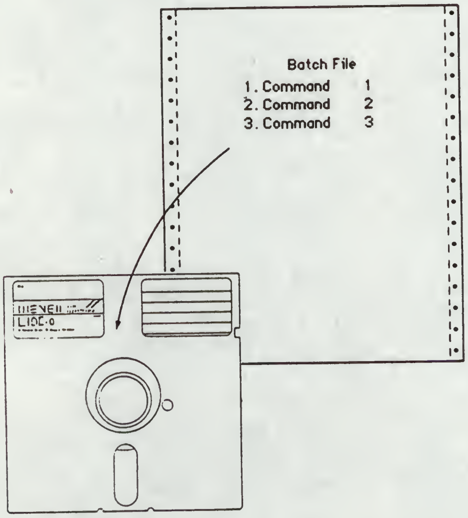


Figure 2.1. Batch File

You can create a batch file by using EDLIN, the line editor, or by typing the Copy command. Refer to the "How to Create an AUTOEXEC.BAT File" section later in this chapter for more information on using the Copy command to create a batch file.

Two MS-DOS commands are used only in batch files: Rem and Pause. Rem allows you to include comments in your batch files without these lines being executed as commands. Pause prompts you with an optional message and permits you to either continue or stop the batch process at a given point. Rem and Pause are described in detail in Chapter 3.

Batch processing is useful if you want to execute several MS-DOS commands with one batch command. For example, a batch file to format and check a new disk might look like this:

```
     1:  Rem  This is a file to check new disks
     2:  Rem  It is named NEWDISK.BAT
     3:  Pause  Insert new disk in drive B:
     4:  Format b:
     5:  Chkdsk b:
```

To execute this .BAT file after it has been entered and saved as NEWDISK.BAT, simply type the filename without the .BAT extension:

```
     newdisk
```

The result is the same as if the lines in the .BAT file were entered from the keyboard as individual commands.

Figure 2.2 shows the three steps used to write, save, and execute an MS-DOS batch file.

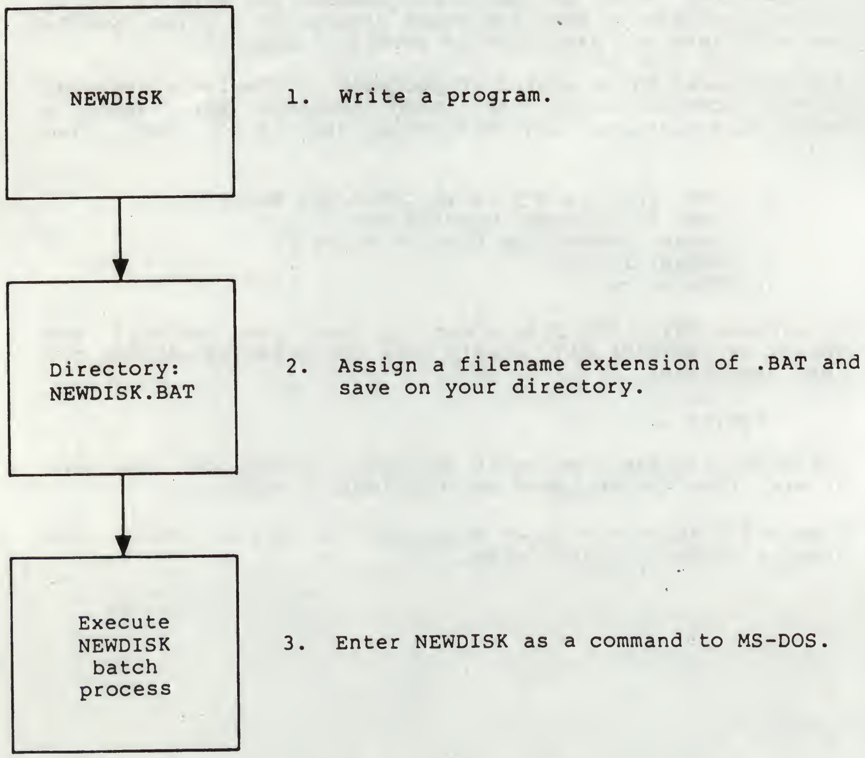

NEWDISK
1. Write a program.
Directory: NEWDISK.BAT
2. Assign a filename extension of .BAT and save on your directory.
Execute NEWDISK batch process
3. Enter NEWDISK as a command to MS-DOS.

Figure 2.2. MS-DOS Batch File Steps

The following list contains information that you should read before you run a batch process with MS-DOS:

1. Only the filename should be entered to execute the batch file. Do not type the filename extension.

2. If you press Control-C when the batch file is running, this prompt appears:

   ```
   Terminate batch job (Y/N)?
   ```

   If you press Y, the remaining commands in the batch file are ignored and the MS-DOS prompt (usually A>) appears.

   If you press N, the current command aborts and batch processing continues with the next command in the file.

3. If you remove the disk that contains a batch file being run, MS-DOS will prompt you to insert it again before the next command can be read.

4. The last command in a batch file can be the name of another batch file. This allows you to call one batch file from another when the first is finished.

5. You can redirect output in a batch file using the > and < symbols. However, you cannot use piping (the | symbol) in a batch file. Refer to section 2.5.1, "How to Redirect Your Output," and section 2.5.3, "Command Piping," for more information.

2.7 THE AUTOEXEC.BAT FILE

An AUTOEXEC.BAT file allows you to automatically execute programs when you start MS-DOS. This is useful when you want to run a specific package (for example, Microsoft Multiplan(R)) under MS-DOS, and when you want MS-DOS to execute a batch program each time you start your computer. You can avoid loading two separate disks to perform either of these tasks by using an AUTOEXEC.BAT file.

When you start your computer, MS-DOS searches the disk for a file named AUTOEXEC.BAT. The AUTOEXEC.BAT file is a batch file that is automatically executed each time you start the system. It must be located in the root directory.

If MS-DOS finds the AUTOEXEC.BAT file, the file is immediately executed and the date and time prompts are bypassed.

If MS-DOS does not find an AUTOEXEC.BAT file when you first load the MS-DOS disk, then the date and time prompts appear. Figure 2.3 shows how MS-DOS uses the AUTOEXEC.BAT file.

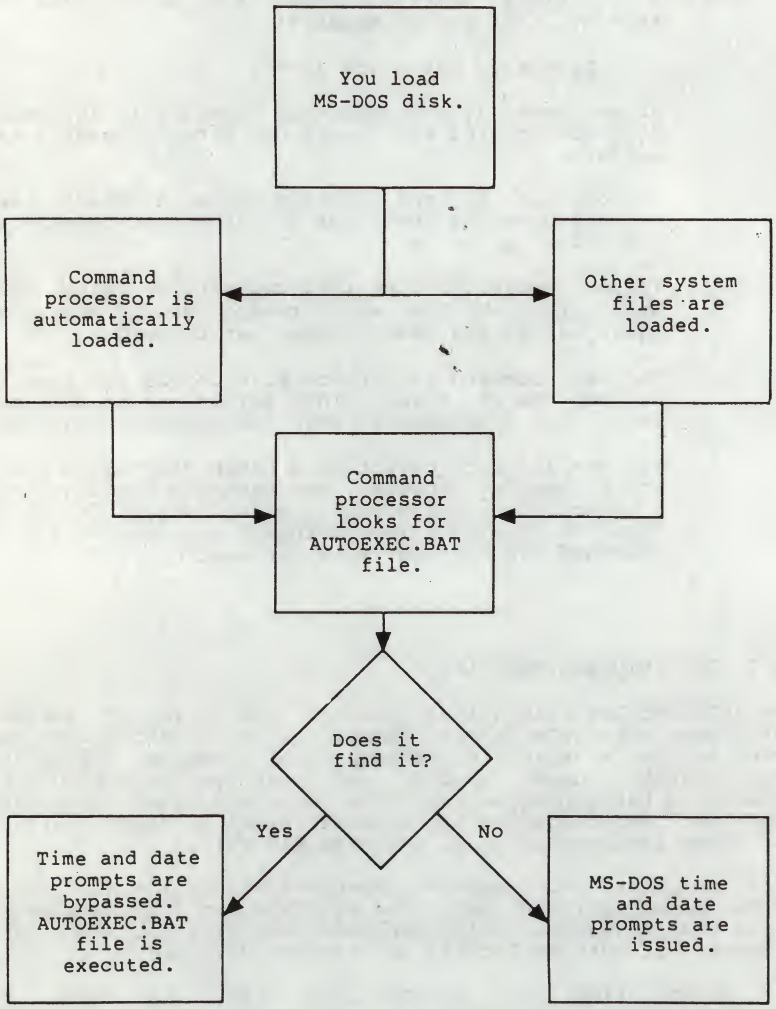

You load MS-DOS disk.
Command processor is automatically loaded.
Other system files are loaded.
Command processor looks for AUTOEXEC.BAT file.
Does it find it?
Yes
No
Time and date prompts are bypassed. AUTOEXEC.BAT file is executed.
MS-DOS time and date prompts are issued.

Figure 2.3. How MS-DOS Uses the AUTOEXEC.BAT File

2.7.1 How to Create an AUTOEXEC.BAT File

If, for example, you want to automatically load BASIC and run a program called MENU each time you start MS-DOS, you could create an AUTOEXEC.BAT file as follows:

1. Type:

   ```
   copy con autoexec.bat
   ```

   This statement tells MS-DOS to copy the information from the console (keyboard) into the AUTOEXEC.BAT file. Note that the AUTOEXEC.BAT file <u>must</u> be created in the root directory of your MS-DOS disk.

2. Now type:

   ```
   basic menu
   ```

   This statement goes into the AUTOEXEC.BAT file. It tells MS-DOS to load BASIC and run the MENU program whenever MS-DOS is started.

3. Press Control-Z and then press the Return key to put the command BASIC MENU in the AUTOEXEC.BAT file.

4. The MENU program will now run automatically whenever you start MS-DOS.

To run your own BASIC program, type the name of your program in place of MENU in the second line of the example. You can enter any MS-DOS command or series of commands in the AUTOEXEC.BAT file.

Helpful Hint

If you use an AUTOEXEC.BAT file, MS-DOS will not prompt you for a current date and time unless you include the Date and Time commands in the AUTOEXEC.BAT file. We recommend that you include these two commands in your AUTOEXEC.BAT file, since MS-DOS uses this information to keep your directory current. Refer to Chapter 3, "MS-DOS Commands," for more information on the Date and Time commands.

2.8 HOW TO CREATE A BATCH FILE WITH REPLACEABLE PARAMETERS

There may be times when you want to create a program and run it with different sets of data. These data may be stored in various MS-DOS files.

When used in MS-DOS commands, a parameter is an option that you define. With MS-DOS, you can create a batch (.BAT) file with dummy (replaceable) parameters. These parameters, named %0-%9, can be replaced by values supplied when the batch file executes.

For example, when you type the command line:

```
copy con newfile.bat
```

the next lines you type are copied from the console to a file named NEWFILE.BAT on the default drive:

```
del  %3.doc
copy %1.doc + %2.doc %3.doc
print %3.doc
```

Now, press Control-Z and then press the Return key. MS-DOS responds with this message:

```
1 File(s) copied
A>_
```

The file NEWFILE.BAT, which consists of three commands, now resides on the disk in the default drive.

The dummy parameters %1 and %2 are replaced sequentially by the parameters you supply when you execute the file. If you use the dummy parameter %0, it is always replaced by the drive name, if specified, and the filename of the batch file (for example, NEWFILE).

NOTES:

1. Up to 10 dummy parameters (%0-%9) can be specified. Refer to the MS-DOS Shift command in Chapter 3 if you want to specify more than 10 parameters.

2. If you use the percent sign as part of a filename within a batch file, you must type it twice. For example, to specify the file ABC%.EXE, you must type it as ABC%%.EXE in the batch file.

2.8.1 How to Run a Batch File

To execute the batch file NEWFILE.BAT and to specify the parameters that will replace the dummy parameters, you must enter the batch filename (without its extension) followed by the parameters you want MS-DOS to substitute for %1, %2, and %3.

Remember that the file NEWFILE.BAT consists of 3 lines:

```
del %3.doc
copy %1.doc + %2.doc %3.doc
print %3.doc
```

To execute the NEWFILE batch process, type

```
newfile a:memo1 b:memo2 b:bigfile
```

and press Return. a:memo1 is substituted for %1, b:memo2 for %2, and b:bigfile for %3.

The result is the same as if you had typed each of the commands in NEWFILE with their parameters, as follows:

```
del b:bigfile
copy a:memo1 + b:memo2 b:bigfile
print b:bigfile
```

The following table shows how MS-DOS replaces each of the above parameters:

BATCH FILENAME	PARAMETER1 (%1)	PARAMETER2 (%2)	PARAMETER3 (%3)
newfile	memo1.doc	b:memo2.doc	b:bigfile

Remember that if you specify the dummy parameter %0 in your batch file, it is always replaced by the drive name (if specified) and the filename of the batch file. If you do not need to refer to the batch file, start your dummy parameters at number %1.

2.9 SUMMARY OF COMMANDS IN THIS CHAPTER

COMMAND	PURPOSE	EXAMPLE
Rem	Adds comment line for batch files	rem This is a test file
Pause	Suspends execution of a batch file	pause Insert blank disk
Find	Searches for string of text in a specified file	find who newfile.txt
More	Pages through a file 23 lines at a time	type memo.txt | more
Sort	Sorts text	sort names.fil

2.10 WHAT'S NEXT?

The next chapter describes each MS-DOS command in detail.

Chapter 3
MS-DOS Commands

CHAPTER 3

MS-DOS COMMANDS

3.1 ABOUT COMMANDS

You should keep the following points in mind as you format MS-DOS commands:

1. You can type commands in any combination of uppercase or lowercase letters; MS-DOS changes them to uppercase letters.

2. Supply the text for any items enclosed in angle brackets (< >). For example, you should type the name of your file when <filename> is shown in the format.

3. Items in square brackets ([ ]) are optional. If you want to include optional information, do not include the square brackets, only the information within the brackets.

4. An ellipsis (...) means that you can repeat an item as many times as necessary.

5. For some commands, the syntax is too long to fit onto one line, so it is split into two (or more) lines. You, however, must give the command on one line.

6. You must include all punctuation where shown (with the exception of square brackets and angle brackets), such as commas, equal signs, question marks, colons, or slashes.

7. Unlike previous versions of MS-DOS, you can specify a pathname to run a specific command or program. For example, if you move the Assign program to a directory named \bin, you can type:

   ```
   \bin\assign c=e
   ```

8. Many commands manipulate "strings" of text. A "string" is a group of characters; it can include letters, numbers, spaces, and all other characters. Searching for a particular word in a file is a common use of a string.

3.2 ABOUT THIS CHAPTER

The commands in this chapter are divided into two sections:

1. Common MS-DOS commands

2. Batch commands

Commands are either external or internal. Each external command resides on disk as a file. Internal commands are included in a file named COMMAND.COM on the MS-DOS disk. External and internal commands are shown by the following symbols:

$\boxed{I}$ This shows that the command is internal

$\boxed{E}$ This shows that the command is external.

Some of the MS-DOS commands do not work over a computer network. If you try to use these commands, MS-DOS displays this error message:

```
Cannot <command> to a network device
```

where <command> is the name of the command you typed. The commands that do not work over a network (on a shared or remote device) are:

Chkdsk
Diskcomp
Diskcopy
Format
Label
Recover
Subst
Sys

If the command does not work over a network (on a shared or remote device), you will see this symbol with the command description:

NET This shows that the command does not work over a network.

3.3 MS-DOS COMMANDS

Note

If you have only one disk drive, or if you have installed logical drives, refer to Appendix A for instructions before running the following commands.

The following MS-DOS commands are described in this chapter. Note that synonyms for commands are in parentheses.

Append	Sets a search path for data files.
Assign	Assigns a drive letter to a different drive.
Attrib	Sets or displays attributes of a file.
Backup	Backs up files from a hard disk.
Break	Sets Control-C check.
Chdir	Changes directories; prints working directory (CD).
Chkdsk	Scans the directory of the default or designated drive and checks for consistency.
Cls	Clears the screen.
Command	Processes internal MS-DOS commands.
Copy	Copies the file(s) specified.
Ctty	Changes the console TTY.
Date	Displays and sets the date.
Del	Deletes the file(s) specified (Erase).

Dir	Lists requested directory entries.
Diskcomp	Compares disks.
Diskcopy	Makes a copy of a disk.
Exe2bin	Converts executable files to binary format.
Exit	Exits command processor and returns to previous level.
Find	Searches for a constant string of text.
Format	Formats a disk to receive MS-DOS files.
Graftabl	Loads a table of graphics characters.
Graphics	Prepares MS-DOS for printing graphics.
Join	Joins a disk drive to a pathname.
Label	Labels disks.
Mkdir	Makes a directory (MD).
Mode	Modifies screen, communications port, and printer port parameters.
More	Displays output one screen at a time.
Path	Sets a command search path.
Print	Prints files.
Prompt	Assigns a default prompt.
Recover	Recovers a bad disk or file.
Ren	Renames first file as second file (Rename).
Replace	Replaces previous versions of files.
Restore	Restores backed up files.
Rmdir	Removes a directory (RD).
Set	Sets one string value to another in the environment, or displays the environment.
Share	Installs file sharing and locking.
Sort	Sorts data forward or backward.
Subst	Substitutes a string for a pathname.

Sys	Transfers MS-DOS system files from one drive to the drive specified.
Time	Displays and sets the time.
Tree	Displays directory and file names.
Type	Displays the contents of a file.
Ver	Prints MS-DOS version number.
Verify	Verifies all writes to a disk.
Vol	Displays the volume ID.
Xcopy	Copies files and subdirectories.

Batch Commands

Echo	Turns batch file echo feature on/off or displays setting.
For	Batch command extension.
Goto	Batch command extension.
If	Batch command extension.
Pause	Pauses for input in a batch file.
Rem	Displays a comment in a batch file.
Shift	Increases number of replaceable parameters in batch process.

These commands will be described in detail in the following pages.

E

NAME

Append

PURPOSE

Sets a search path for data files.

SYNTAX

```
append
[<drive:>][<path>][;[<drive:>][<path>]...]
```

COMMENTS

The <path> parameter specifies the directory that MS-DOS searches for a data file.

You can specify more than one path to search by separating each path with a semicolon (;).

If you use the Append command with no options, MS-DOS displays the current data path. If you use the command:

```
append ;
```

MS-DOS sets the NUL data path. This means that MS-DOS searches only the working directory for data files.

Append searches the data path for all files, regardless of the their file extensions, only with the following MS-DOS system calls:

Code	Function
0FH	Open File (FCB)
23H	Get File Size
3DH	Open Handle

You can use the Append command across a network to locate remote data files.

Note

If you are using both the MS-DOS Assign and Append commands, you must type the Append command before Assign.

Example:

Suppose you want to access data files in a directory called LETTERS on drive B, and in a directory called REPORTS on drive A. To do this, use the command:

```
append b:letters;a:reports
```

E

NAME

Assign

PURPOSE

Assigns a drive letter to a different drive.

SYNTAX

```
assign[[x]=[y]]
```

COMMENTS

The equal sign (=) is optional.

x is the drive to which reads and writes are currently sent.

y is the drive to which you want reads and writes sent.

You do not have to type the colon after the drive letters x and y.

To reset all drives back to their original assignments, type:

```
assign
```

and press the Return key.

The Assign comand allows you to perform disk operations on drives other than A and B for programs that only use those two drives. The command

```
assign a=c b=c
```

enables you to use drives other than A and B, such as a hard disk (C). All references to drives A and B now go to drive C.

Note

You should not use the Assign command with the Backup or Print commands or during normal use of MS-DOS. This is because the Assign command hides the true device type from commands that require actual drive information. The Format and Diskcopy programs ignore any drive reassignments.

MESSAGES

Incorrect parameter

One of the options you specified is wrong.

E

NAME

Attrib

PURPOSE

Sets or resets the read-only and archive attributes of a file; displays the attributes of a file.

SYNTAX

```
attrib[+r|-r] [+a|-a] [<drive:>] <pathname>
```

COMMENTS

+R sets the read-only attribute of a file.

-R disables read-only mode.

+A sets the archive attribute of a file.

-A clears the archive attribute of a file.

[<drive:>]<pathname> specifies the file you want to reference.

If an application opens a file with read and write permission, Atrib forces read-only mode to allow file sharing over a network.

The Backup, Restore, and Xcopy (Mcopy) commands use the archive attribute to control a selective Backup/Restore/Xcopy on files that have been modified. You can use the +A and -A options to select files that you want to back up with the Backup /m switch or copy with the Xcopy (or Mcopy) /m switch.

To display the attribute of a specific file, type:

```
attrib <pathname>
```

and press the Return key.

The wildcard characters *.* can be used to print the attributes of all files on a specific drive.

Example:

The following example makes the file named MYFILE.TXT read-only:

```
attrib +r myfile.txt
```

E

NAME

Backup

PURPOSE

Backs up one or more files from one disk to another.

SYNTAX

```
backup#[<drive:>][<pathname>][<drive:>][/s][/m]
[/a][/d:<date>] [/t:<time>] [/L:<filename>]
```

COMMENTS

The Backup command can back up files on disks of different media. For example, you can back up files from:

Hard disk to floppy disk
Floppy disk to floppy disk
Floppy disk to hard disk
Hard disk to hard disk

Backup backs up files from one floppy disk to another, even if the disks have a different number of sides or sectors.

The first option that you specify is <drive:>, the drive that contains the files that you want to back up. The second <drive:> is the backup disk drive. Unless otherwise specified, the old files on a backup disk are erased before new files are added to it.

If the target drive is a floppy disk, Backup places the backed up files in the root directory. If the target drive is a hard disk, Backup places the backed up files in a subdirectory called BACKUP. You must format new disks before Backup can use them.

This backup program and the one supplied by IBM(R) are compatible. File and disk formats are the same as those used by the IBM Backup program unless you specify the /p or /L switches. You may not be able to use the IBM Restore program to restore files that were backed up with the /p or /L switch.

You can use the following switches with Backup:

/s	Back up subdirectories also.
/m	Back up only those files that have changed since the last backup.
/a	Add the files to be backed up to those already on the backup disk. Do not erase old files on the backup disk.
/d	Back up only those files that were last modified at or after a certain date.
/p	Pack as many files as possible onto each disk. Create a subdirectory on the disk if that is the only way to fill the backup disk. (WARNING: IBM Backup/ Restore compatibility may be lost if you use this switch.)
/t	Back up only those files that were last modified at or after a certain time.
/L	Make a backup log entry in the specified file. If you do not give a filename, Backup places the entries in a file called BACKUP.LOG in the root directory of the files being backed up. The first line in each entry contains: [date time] where date and time are the backup dates and times. Each subsequent line in an entry corresponds to one of the files that you backed up. These lines consist of the filename and the number of the floppy disk that contains the file. You can use this information when you need to restore a particular file from a floppy disk. If the backup log already exists, Backup appends the entry to the end of the file. (WARNING: IBM Backup/ Restore compatibility may be lost if you use this switch.)

Backup displays the name of each file as it is backed up. You should label and number each backup disk consecutively to help you restore the files properly with the Restore command.

If you are sharing files, you can backup only the files that you have access to.

Note

You should not use the Backup command if the drive you are backing up has been assigned, joined, or substituted. If you do, you may not be able to restore the files with the Restore command.

The Backup program returns the following errorlevel codes:

0	Normal completion
1	No files were found to back up
2	Some files not backed up due to sharing conflicts
3	Terminated by user
4	Terminated due to error

You can use the batch processing If command to perform error processing based on the errorlevel returned by Backup.

NAME

Break

PURPOSE

Sets Control-C check.

SYNTAX

```
break [ON]

 or

break [OFF]
```

COMMENTS

Depending on the program you are running, Control-C may be used to stop an activity (for example, to stop sorting a file). Normally, MS-DOS checks to see if Control-C has been typed while it is reading from the keyboard, writing to the screen, or to a printer. Setting Break to "on" allows Control-C checking to be extended to other functions such as disk reads and writes.

To check for Control-C during screen, keyboard, and printer reads and writes only, set Break to OFF.

To find out how Break is currently set, type:

```
break
```

and press the Return key.

NAME

Chdir (Change Directory)

SYNONYM

cd

PURPOSE

Changes directory to a different path; displays working directory.

SYNTAX

```
chdir [<pathname>]
```

COMMENTS

If your working directory is \BIN\USER\JOE and you want to change your path to another directory (such as \BIN\USER\JOE\FORMS), type:

```
chdir \bin\user\joe\forms
```

and press the Return key.

MS-DOS will put you in the new directory. A shorthand notation for the Chdir command is also available:

```
chdir ..
```

This command always puts you in the <u>parent</u> directory of your working directory.

Chdir used without a pathname displays your working directory. If your working directory is \BIN\USER\JOE on drive B, and you type:

```
chdir
```

then press the Return key, MS-DOS displays:

```
b:\bin\user\joe
```

Using the Chdir command in this way is useful when you need to know the name of your working directory.

The command:

```
chdir b:
```

displays the working directory on drive B.

NAME

Chkdsk (Check Disk)

NET E

PURPOSE

Scans the disk in the specified drive and checks it for errors.

SYNTAX

```
chkdsk [<drive:>] [<pathname>] [/f] [/v]
```

COMMENTS

Chkdsk should be run occasionally on each disk to check for errors. If any errors are found, Chkdsk displays the appropriate error messages, if any, followed by a status report.

A sample status report looks like this:

```
Volume DOSDISK       created Jan 12, 1986 3:21p

     362496  bytes total disk space
      38912  bytes in 2 hidden files
       1024  bytes in 2 directories
     282624  bytes in 42 user files
      39936  bytes available on disk

     524288  bytes total memory
     487248  bytes free
```

Chkdsk does not correct any errors found in your directory unless you specify the /f (fix) switch. If you specify the /v switch, Chkdsk displays messages while it is running.

If you type a filename after Chkdsk, MS-DOS displays a status report for the disk and for the individual file.

You can redirect the output from Chkdsk to a file if you want to save the status report for future use. Simply type:

```
chkdsk a:>filename
```

The errors are sent to the specified filename. Do not use the /f switch if you redirect Chkdsk output.

MESSAGES

Chkdsk may display the following messages:

(.)(..) does not exist
: This is an informational message from Chkdsk. This message indicates that either the "." or the ".." directory entry is invalid.

(filename) contains
non-contiguous blocks
: The file or files you named are not written contiguously on the disk.

All specified file(s) are contiguous
: The file or files you named are all written sequentially on disk.

The following errors are corrected automatically if you specify the /f switch:

Allocation error, size adjusted
: An invalid cluster number was found in the FAT. The file was truncated at the end of the last valid cluster.

Entry has a bad attribute (or size or link)
: This message may be preceded by one or two periods to indicate which subdirectory is invalid. If you have specified the /f switch, Chkdsk attempts to correct the error.

Errors found, F parameter not specified
Corrections will not be written to disk
: You must specify the /f switch if you want the errors corrected by Chkdsk.

First cluster number is invalid
Entry truncated
: The file directory entry contains an invalid pointer to the data area. If you specified the /f switch, the file is truncated to a zero-length file.

X lost clusters found in y chains
Convert lost chains to files (Y/N)?

If you respond Y to this prompt, and if you specified the /f switch, Chkdsk creates a file in the root directory for you to resolve this problem (files created by Chkdsk are named FILEnnnn.CHK). If you did not specify the /f switch, Chkdsk does nothing.

If you respond N to this prompt and have specified the /f switch, Chkdsk displays:

```
X bytes disk space freed
```

If you respond N to this prompt and have not specified the /f switch, Chkdsk displays:

```
X bytes disk space would be freed
```

MS-DOS cannot correct the following errors returned by Chkdsk, even if you specified the /f switch. You must take action to correct the situation:

(filename) is cross-linked on cluster

Make a copy of the file you want to keep, and then delete both files that are cross-linked on the same cluster.

Cannot Chdir to (filename)
Tree past this point not processed

Chkdsk is traveling the tree structure of the directory and is unable to proceed to the specified directory. All subdirectories and files under this directory are not verified.

Cannot Chdir to root
Processing cannot continue

Chkdsk is traveling the tree structure of the directory and is unable to return to the root directory. Chkdsk is not able to continue checking the remaining subdirectories to the root. Try to restart MS-DOS. If this error persists, the disk is unusable.

Cannot recover . entry, processing continued

The "." entry (working directory) is defective.

Cannot recover .. entry
: The ".." (parent directory) entry is defective.

Directory is totally empty, no . or ..
: The specified directory does not contain references to working and parent directories. Delete the specified directory and recreate it.

Disk error reading FAT
: One of your File Allocation Tables has a defective sector in it. MS-DOS will automatically use the other FAT. It is a good idea to copy all your files onto another disk.

Disk error writing FAT
: One of your File Allocation Tables has a defective sector in it. MS-DOS will automatically use the other FAT. It is a good idea to copy all your files onto another disk.

Incorrect DOS version
: You cannot run Chkdsk on versions of MS-DOS before 3.0.

Insufficient memory
Processing cannot continue
: There is not enough memory in your machine for Chkdsk to process this disk. You must obtain more memory to run Chkdsk.

Insufficient room in root directory
Erase files in root and repeat Chkdsk
: Chkdsk cannot run until you delete files in the root directory.

Invalid drive specification
: Specify a valid drive.

Invalid parameter
: One of the switches that you have specified is wrong.

Invalid sub-directory entry
: The subdirectory that you have specified either does not exist or is invalid. Check to see that you have entered the subdirectory name correctly.

Invalid working directory
Processing cannot continue

> Your disk is bad. Replace the disk or make a copy from your backup system disk.

Probable non-DOS disk
Continue (Y/N)?

> The disk you are using is a non-DOS disk. You must indicate whether or not you want Chkdsk to continue processing. This error message usually means that the File Allocation Table (FAT) is bad.

Unrecoverable error in directory
Convert directory to file (Y/N)?

> If you respond Y to this prompt, Chkdsk converts the bad directory into a file. You can then delete it.
>
> If you respond N to this prompt, you may not be able to write to or read from the bad directory.

I

NAME

Cls

PURPOSE

Clears the terminal screen.

SYNTAX

```
cls
```

COMMENTS

The Cls command causes MS-DOS to send the ANSI escape sequence ESC[2J (which clears your screen) to your console.

NAME

Command

PURPOSE

Starts the command processor.

SYNTAX

```
command [<drive:><pathname>][<cttydev>][/p]
[/c <string>] [/e:<nnnnn>]
```

COMMENTS

This command starts a new command processor (the MS-DOS program that contains all internal commands).

The command processor is loaded into memory in two parts: the transient part and the resident part. Some application programs write over the transient part of COMMAND.COM when they run. When this happens, the resident part of the command processor looks for the COMMAND.COM file on disk so it can reload the transient part.

The <drive:><pathname> options tell the command processor where to look for the COMMAND.COM file if it needs to reload the transient part into memory.

<cttydev> allows you to specify a different device (such as aux) for input and output. See the Ctty command in this chapter for more information.

The /e switch specifies the environment size, where <nnnnn> is the size in bytes. The size may range between 128 and 32768 bytes. The default value is 128 bytes.

If <nnnnn> is less than 128 bytes, MS-DOS defaults to 128 bytes and gives the message:

```
Invalid environment size specified
```

If <nnnnn> is greater than 32768 bytes, MS-DOS gives the same message, but defaults to 32768 bytes.

The /p switch tells COMMAND.COM not to exit to any higher level.

The /c switch, if used, should be the last switch in the command. It tells the command processor to execute the command or commands specified by <string> and then return.

Example:

```
command /c chkdsk b:
```

This example tells the command processor to:

1. Start a new command processor under the current program.

2. Run the command "chkdsk b:"

3. Return to the first command processor.

Refer to the sample CONFIG.SYS file in Appendix D, "How to Configure Your System," to see how a pathname and the /p switch are used with Command.

NAME

Copy

PURPOSE

Copies one or more files to another disk. If you prefer, you can give the copies different names. This command also copies files on the same disk and appends files.

SYNTAX

```
copy [<drive:>]<pathname> [<drive:>]
[<pathname>][/v] [/a] /b] (to copy)

copy <pathname> + <pathname> ... <pathname>
(to append)
```

COMMENTS

<u>To copy files</u>:

> **Note**
>
> If the source and target files are both in the working directory, you may use filenames, rather than complete pathnames.

If the second pathname is not given, the copy will be on the default drive and will have the same name as the original file (first pathname). If the first pathname is on the default drive and if you do not specify the second pathname, the Copy command will quit. (Copying files to themselves is not allowed.) MS-DOS will display the error message:

```
File cannot be copied onto itself
0 File(s) copied
```

The second option may take three forms:

1. If the second option is a drive name only, the original file is copied with the original filename to the designated drive. (For example, "copy memo.doc b:" will make a copy on drive B named MEMO.DOC.)

2. If the second option is a filename only, the original file is copied to a file on the default drive with the filename specified. (For example, "copy memo.doc letter.doc" will make a copy of MEMO.DOC, name it LETTER.DOC, and place it on the default drive.)

3. If the second option includes a drive name, the original file is copied to a file on the drive specified. (For example, "copy memo.doc b:memo.doc" will make a copy of memo.doc on the default drive, name the copy MEMO.DOC, and place the copy on the disk in drive B.)

The /v switch causes MS-DOS to verify that the sectors written on the target disk are recorded properly. Although there are rarely recording errors when you run Copy, you can verify that critical data has been correctly recorded. This option causes the Copy command to run more slowly because MS-DOS must check each entry recorded on the disk. An error is displayed if a write is not verified.

The /a and /b switches mean that the files being processed are ASCII or binary files. Each switch applies to the filename preceding it, and to all remaining filenames in the command, until another /a or /b is encountered.

The following discussion applies to the /a and /b switches:

When used with a source filename:

/a	Causes the file to be treated as an ASCII (text) file. Data in the file is copied up to but not including the first end-of-file mark (in EDLIN, this is Control-Z). The remainder of the file is not copied.
/b	Causes the entire file to be copied, including any end-of-file mark.

Examples:

```
copy memo.txt /a letter.txt

copy report.asm /b report2.asm
```

When used with a destination filename:

/a	Causes an end-of-file character to be added as the last character of the file.
/b	Causes no end-of-file character to be added.

When you are appending files, the default switch is always /a.

Examples:

```
copy memo.txt letter.txt /a

copy report.asm report2.asm /b
```

To append files:

The Copy command also allows you to append files. Simply list any number of files as options to Copy, separated by +, then specify a target file to send the combined files to.

For example:

```
copy intro.rpt + body.rpt + b:sum.rpt report
```

This command combines files named INTRO.RPT, BODY.RPT, and SUM.RPT (on drive B) and places them in the file on the default drive called REPORT.

To combine several files using wildcards into one file, you could type:

```
copy *.lst combin.prn
```

This command takes all files with a filename extension of .LST and combines them into a file named COMBIN.PRN.

In the following example, for each file found matching *.LST, that file is combined with the corresponding .REF file. The result is a file with the same filename but with the extension .PRN. Thus, FILE1.LST will be combined with FILE1.REF to form FILE1.PRN; then XYZ.LST with XYZ.REF to form XYZ.PRN; and so on.

```
copy *.lst + *.ref *.prn
```

The following Copy command combines all files matching *.LST, then all files matching *.REF, into one file named COMBIN.PRN:

```
copy *.lst + *.ref combin.prn
```

Note

Do not try to append files where one of the source filenames has the same extension as the target. For example, the following command is an error if ALL.LST already exists:

```
copy  *.lst  all.lst
```

The error would not be detected, however, until ALL.LST is appended. At this point ALL.LST could have already been destroyed.

Copy compares the filename of the input file with the filename of the target. If they are the same, that one input file is skipped, and the error message "Content of destination lost before copy" is printed. Further joining proceeds normally. This allows "summing" files, as in this example:

```
copy  all.lst + *.lst
```

This command appends all *.LST files, except ALL.LST itself, to ALL.LST. This command will not produce an error message.

MESSAGES

Cannot do binary reads from a device

The copy cannot be done in binary mode when you are copying from a device. Remove the /b switch or specify an ASCII copy with the /a switch.

Content of destination lost before copy

A file to be used as a source file has been overwritten prior to completion of the copy.

Example:

```
copy x + y y
```

which destroys Y before it can be copied.

NAME

Ctty

PURPOSE

Allows you to change the device from which you issue commands. (In this command, the letters tty represent the console; that is, your keyboard.)

SYNTAX

```
ctty <device>
```

COMMENTS

The <device> is the device from which you are giving commands to MS-DOS. This command is useful if you want to change the device on which you are working. The command

```
ctty aux
```

moves all command I/O (input/output) from the current device (the console) to an AUX port such as another terminal. The command

```
ctty con
```

moves I/O back to the console. Refer to the "Illegal Filenames" section of Chapter 1, "Files and Directories," for a list of valid device names to use with the Ctty command.

Note

There are many programs that do not use MS-DOS for input, output, or both. These programs do input directly to the hardware on your computer. The Ctty command will have no effect on these programs. Ctty will only affect programs that use MS-DOS.

I

NAME

Date

PURPOSE

Enter or change the date known to the system. This date will be recorded in the directory for any files you create or change.

You can change the date from your terminal or from a batch file. (MS-DOS does not display a prompt for the date if you use an AUTOEXEC.BAT file, so you may want to include a Date command in that file.)

SYNTAX

```
date [<mm>-<dd>-<yy>]
```

COMMENTS

If you type:

```
date
```

Date responds with the message:

```
Current date is (weekday) (mm)-(dd)-(yy)
Enter new date (mm-dd-yy):_
```

Press the Return key if you do not want to change the date shown.

You can also type a particular date after the Date command, as in:

```
date 3-9-86
```

In this case, the "Enter new date:" prompt does not appear after you have pressed Return.

Use only numbers when you type the date; allowed numbers are:

```
<mm> = 1-12
<dd> = 1-31
<yy> = 80-99 (or 1980-2099)
```

The date, month, and year entries may be separated by hyphens (-) or slashes (/). MS-DOS is programmed to change months and years correctly, whether the month has 31, 30, 29, or 28 days. MS-DOS handles leap years, too.

It is possible to change the mm-dd-yy format in which the date is displayed and entered. The Country command in the CONFIG.SYS file allows you to change the date format to the European standard dd-mm-yy. Refer to Appendix D, "How to Configure Your System," for more information on the CONFIG.SYS file.

MESSAGES

```
Invalid date
Enter new date (mm-dd-yy):_
```

If the numbers or separators are not valid, Date displays this message. Enter a valid date.

I

NAME
Del (Delete)

SYNONYM
Erase

PURPOSE
Deletes all files with the designated file specification.

SYNTAX
```
del [<drive:>] <pathname>
```

COMMENTS
If the pathname is *.*, the prompt "Are you sure?" appears. If a Y is typed as a response, then all files on the disk are deleted.

NAME

Dir (Directory)

PURPOSE

Lists the files in a directory.

SYNTAX

```
dir [<drive:>][<pathname>][/p][/w]
```

COMMENTS

If you just type Dir, all directory entries on the default drive are listed. If you include a drive name, such as "dir b:", all entries on the disk in the specified drive are listed. If only a filename is entered with no extension ("dir filename"), then all files with the designated filename on the disk in the default drive are listed. When you type a filename with a drive letter (for example, "dir b:filename.ext"), all files with the filename specified on the disk in the drive specified are displayed. In all cases, files are listed with their size in bytes and with the time and date of their last modification. The wildcards ? and * (question mark and asterisk) may be used in the filename option. Note that the following Dir commands are equivalent:

Command	**Equivalent**
dir	dir *.*
dir filename	dir filename.*
dir .ext	dir *.ext

Two switches may be used with Dir. The /p switch selects Page Mode. With /p, display of the directory pauses after the screen is filled. To resume display of output, press any key.

The /w switch selects wide display. With /w, only filenames are displayed, without other file information. Files are displayed five per line.

Note that if the Country command in the CONFIG.SYS file is set to a country other than the U.S., the directory date and time formats will differ. Refer to Appendix D, "How to Configure Your System," for more information on the CONFIG.SYS file.

Note

The Dir command cannot list the directory if you specify a pathname of more than 64 characters (including the drive name). Therefore, you may need to change directories if you want to list files in deep subdirectories.

NAME

Diskcomp

PURPOSE

Compares the contents of the disk in the source drive to the disk in the target drive.

SYNTAX

```
diskcomp [<drive:>] [<drive:>] [/1] [/8]
```

COMMENTS

The first <drive:> option specifies the drive that contains the source disk, and the second <drive:> option specifies the drive that contains the target disk.

If you specify only one drive, Diskcomp uses the default drive as the target drive. If you specify the same drive as the source and target, Diskcomp does a comparison using one drive, and prompts you to insert the disks as appropriate.

The /1 switch causes Diskcomp to compare just the first side of the disk, even if the disks and drives that you are using are double-sided.

The /8 switch causes Diskcomp to compare just the first 8 sectors per track, even if the disks contain 9 or 15 sectors per track.

Diskcomp performs a track-by-track comparison of the disks. It automatically determines the number of sides and sectors per track based on the format of the source disk. If the target disk is not the same type as the disk in the source drive, Diskcomp displays the message:

```
Drive types (double, single-sided) or
diskette types not compatible
```

If all of the tracks are the same, Diskcomp displays the message:

```
Diskettes compare OK
```

If the tracks are not the same, Diskcomp displays a "Compare error" message that includes the track and side number where it found the mismatch.

When Diskcomp completes the comparison, it prompts:

```
Compare more diskettes (Y/N)?_
```

If you type Y, Diskcomp prompts you to insert the proper disks and does the next comparison. If you type N, Diskcomp ends. If you end Diskcomp and if the disk in the default drive does not contain MS-DOS, Diskcomp prompts you:

```
Insert disk with COMMAND.COM in drive A
and strike any key when ready
```

You cannot use Diskcomp with assigned, joined, or substituted drives. Diskcomp does not work on network drives. If you attempt to use the Diskcomp command with these types of drives, Diskcomp diskpays an error message.

Note

If you are comparing a disk with a backup disk that you made with the Copy command, Diskcomp may give the "Compare error" message, even if the files on the disks are identical. This is because the Copy command duplicates the information, but it doesn't necessarily place the information in the same location on the target disk. In this case, you should use the FC utility to compare individual files on the disk.

When correctly written programs exit back to DOS, they return an error code: 0 if no error occurred, or a value greater than zero if there was a problem. This error code can be tested in batch files, and it allows batch programmers to "branch" to an error-handling routine in the batch file.

The Diskcomp command returns the following errorlevel codes:

0 Compared OK
   The disks compared exactly.
1 Did not compare
   The disks were not the same.
2 Control-C error
   The user terminated with Control-C.
3 Hard error
   An unrecoverable read or write error occurred --did not compare.

4 Initialization error
  There is not enough memory--invalid drives or command line syntax.

MESSAGES

Cannot DISKCOMP to or from
a network drive
: One of the drives that you specified is a network device.

Cannot DISKCOMP to or from
an ASSIGNed or SUBSTed drive
: One of the drives that you specified is a drive that you created using the ASSIGN or SUBST command.

Compare another diskette (Y/N)?
: After comparing the disks, Diskcomp asks if you want to compare another disk. Press Y (for Yes) or N (for N).

Compare error on
side <NNN>, track <nnn>
: Diskcomp found an error on side NNN, track nnn.

Compare OK
: Diskcomp displays this message if the disks are identical.

Compare process ended
: Diskcomp displays this message if a fatal error occured during the comparison.

Comparing (tt) tracks
(xx) sectors per track, (y) side(s)
: This message gives the format (of the first disk) that Diskcomp is using to perform the comparison.

DEVICE Support Not Present
: The disk drive does not support MS-DOS 3.2 device control.

Do not specify filename(s)
Command format: DISKCOMP d: d:[/1][/8]
: You specified an incorrect switch or gave a filename in addition to a drive name.

Drive (x:) not ready
Make sure a diskette is inserted into
the drive and the door is closed
: You have not inserted a disk into the specified disk drive.

Drive types or diskette types
not compatible

You cannot compare a single-sided disk with a double-sided disk, or a high-density disk with a low-density disk. You should use FC to compare the files on the disks.

FIRST diskette bad or incompatible

Diskcomp cannot determine the format of the disk that you want to compare.

Incorrect DOS version

You must use the correct version of MS-DOS with this command.

Insert FIRST diskette in drive (x:)
Press any key when ready . . .

Diskcopy displays this message to prompt you to insert the first disk into the specified drive.

Insert SECOND diskette in drive (x:)
Press any key when ready . . .

Diskcopy displays this message to prompt you to insert the second disk into the specified drive.

Insufficient memory

There is not enough memory available to perform the comparison.

Invalid drive specification

You specified an drive that is incorrect or doesn't exist.

Invalid parameter

You specifed a parameter that is incorrect or doesn't exist.

SECOND diskette bad or incompatible

The second disk does not contain the same format as the first disk, or Diskcomp does not recognize the format of the second disk.

Specified drive does not exist
or is non-removable

You must specify the name of a valid floppy drive. Diskcomp cannot compare hard disks.

Unrecoverable read error on drive (x:)

Diskcomp could not read the disk.

NAME

Diskcopy

PURPOSE

Copies the contents of the disk in the source drive to the disk in the target drive.

SYNTAX

```
diskcopy [<drive:>] [<drive:>]
```

COMMENTS

The first option you specify is the source drive. The second option is the target drive.

If the disk in the target drive is unformatted, Diskcopy formats it with the same format that is on the source disk.

You can specify the same drives or you may specify different drives. If the drives are the same, a single-drive copy operation is performed. You are prompted to insert the disks at the appropriate times. Diskcopy waits for you to press any key before continuing.

After copying, Diskcopy prompts:

```
Copy complete
Copy another (Y/N)?_.
```

If you press Y, MS-DOS will prompt you to insert source and target disks, and the next copy is performed on the same drives that you originally specified.

To end the Diskcopy process, press N.

If you omit both options, a single-drive copy operation is performed on the default drive.

If you omit the second option, the default drive is used as the target drive.

Both disks must have the same number of physical sectors and those sectors must be the same size.

Disks that have had a lot of files created and deleted on them become fragmented, because disk space is not allocated sequentially. The first free sector found is the next sector allocated, regardless of its location on the disk.

A fragmented disk can cause slowness due to delays in finding, reading, or writing a file. If this is the case, you must use either the Copy command or the Xcopy command to copy your disk, instead of Diskcopy. Because Copy and Xcopy both copy files sequentially to a disk, the new disk will not be fragmented.

For example:

```
     xcopy a:*.* b:     /s
```

copies all files from the disk in drive A to the disk in drive B.

Diskcopy figures out the number of sides to copy, based on the source drive and disk.

When correctly written programs exit back to DOS, they return an error code: 0 if no error occurred, or a value greater than zero if there was a problem. This error code can be tested in batch files, and it allows batch programmers to "branch" to an error-handling routine in the batch file.

0 Copied Successfully
  The last Diskcopy was completed with no errors.

1 Non-fatal read/write error
  An unrecoverable but non-fatal read or write error occurred.

2 Control-C error
  The user entered Control-C to terminate Diskcopy.

3 Fatal hard error
  Diskcopy was unable to read the source disk or format the target disk.

4 Initialization error
  There is not enough memory--invalid drives or command line syntax.

MESSAGES

Cannot DISKCOPY to or from a network drive

The user is trying to Diskcopy to or from a network drive.

Cannot DISKCOPY to or from an ASSIGNed or SUBSTed drive

The user is trying to Diskcopy to or from a "virtual" drive that was created by using the SUBST or ASSIGN command.

Copy another diskette (Y/N)?

The user prompted to see if the Diskcopy process was to be repeated on another source/target disk.

Copy not completed

Diskcopy cannot copy the entire disk. Use the Copy command to copy specific files onto another disk.

Copying NN tracks
NN Sectors/Track, NN Side(s)

Diskcopy describes the format of the SOURCE disk.

Disk error while reading drive A:
Abort, Ignore, Retry?

Diskcopy found errors during processing. Refer to Appendix B, "Disk Errors," for information on this message.

Disks must be the same size

You cannot copy the contents of a source disk with a format different from the target disk using Diskcopy. Use the Copy command to copy files onto the disk.

Drive (x:) not ready
Make sure a diskette is inserted into
the drive and the door is closed

The disk is not ready to be read or written to.

Drive types or diskette types not compatible

For example: Attempting to copy to a high density disk in a low density drive.

Formatting while copying

If the target disk is not the same format as the source disk, Diskcopy reformats the target disk.

Incorrect DOS version

Many version 2.x and 3.x utilities will not run on earlier versions of MS-DOS. Some utilities will only run under the exact version of MS-DOS for which they were configured.

Insert [SOURCE|TARGET] diskette in drive D:
Press any key when ready...

This is the prompt to begin the Diskcopy process.

Insufficient memory

Therre is not enough memory to perform the Diskcopy.

Invalid drive specification
Specified drive does not exist
or is non-removable

The user specified an invalid or non-removable drive.

Invalid parameter,
Do not specify filename(s)
Command Format: DISKCOPY (x:) (x:) [/1]

The user specified either an illegal switch or filenames instead of only drive names.

[SOURCE|TARGET] diskette bad or incompatible
Copy process ended

Diskcopy could not read the source diskette, or the target diskette could not be read or formatted.

Target diskette is write protected

The target disk has a write protect tab on it.

Target diskette may be unusable

Unrecoverable read or write errors were encountered while copying to the target disk. The target disk is not an exact copy of the source disk.

Unrecoverable [read|write] error on drive (x:)
side NN, track NN

An error reading/writing to the disk occurred.

Source and target diskettes are not the same format. Cannot do the copy

> You must have the same size and kind of disks to run Diskcopy. For example, you cannot copy from a single-sided disk to a double-sided disk. Reformat the target disk to be of the same type as the source disk or substitute a disk of the same type.

NAME

Exe2bin

PURPOSE

Converts .EXE (executable) files to binary format.

SYNTAX

```
exe2bin [<drive:>]<pathname>
[<drive:>][<pathname>]
```

COMMENTS

This command is useful only if you want to convert .EXE (executable) files to binary format. The file named by <pathname> is the input file. If no extension is specified, it defaults to .EXE. The input file is converted to .BIN file format (memory image of the program) and placed in the output file (second pathname). If you do not specify a drive name, the drive of the input file will be used. If you do not specify an output filename, the input filename will be used. If you do not specify a filename extension in the output filename, the new file will be given an extension of .BIN.

The input file must be in valid .EXE format produced by the linker. The resident, or actual code and data part of the file must be less than 64K. There must be no STACK segment.

Two kinds of conversions are possible, depending on whether the initial CS:IP (Code Segment: Instruction Pointer) is specified in the .EXE file:

1. If CS:IP is not specified in the .EXE file, a pure binary conversion is assumed. If segment fixups are necessary (that is, the program contains instructions requiring segment relocation), you will be prompted for the fixup value. This value is the absolute segment at which the program is to be loaded. The resultant program will be usable only when loaded at the absolute memory address specified by a user application. The command processor will not be able to load the program.

2. If CS:IP is 0000:100H, it is assumed that the file will run as a .COM file with the location pointer set at 100H by the assembler statement ORG; the first 100H bytes of the file are deleted. No segment fixups are allowed, as .COM files must be segment relocatable; that is, they must assume the entry conditions explained in the Microsoft Macro Assembler Manual. Once the conversion is complete, you may rename the output file with a .COM extension. Then the command processor will be able to load and execute the program in the same way as the .COM programs supplied on your MS-DOS disk.

MESSAGES

File cannot be converted
: CS:IP does not meet either of the criteria specified above, or it meets the .COM file criterion but has segment fixups. This message is also displayed if the file is not a valid executable file.

File not found
: The file is not on the disk specified.

Insufficient memory
: There is not enough memory to run Exe2bin.

File creation error
: Exe2bin cannot create the output file. Run Chkdsk to determine if the directory is full, or if some other condition caused the error.

Insufficient disk space
: There is not enough disk space to create a new file.

Fixups needed - base segment (hex):
: The source (.EXE) file contained information indicating that a load segment is required for the file. Specify the absolute segment address at which the finished module is to be located.

File cannot be converted The input file is not
    in the correct format.

WARNING - Read error in EXE file.
Amount read less than size in header
    This is a warning message only. It means
    that the .EXE header in the file is
    inconsistent with the size of the file.

NAME

Exit

PURPOSE

Exits the program COMMAND.COM (the command processor) and returns to a previous level, if one exists.

SYNTAX

```
exit
```

COMMENTS

This command can be used when you are running an application program and want to start the MS-DOS command processor, then return to your program. For example, to look at a directory on drive B while running an application program, you must issue an EXEC of the command interpreter (system call 4BH). The system prompt will appear. You can now type the Dir command and MS-DOS will display the directory listing. When you type "exit", you return to the previous level (your application program).

E

NAME

Find

PURPOSE

Searches for a specific string of text in a file or files.

SYNTAX

```
find [/v] [/c] [/n]
<"string"> [<drive:>] [<pathname>]
```

COMMENTS

Find is a filter that takes as options a string and a series of filenames. It will display all lines that contain a specified string (group of characters) from the files in the command line.

If no filenames are typed, Find takes the input from the screen and displays all lines that contain the specified string.

Switches for Find are:

/v	Causes Find to display all lines not containing the specified string.
/c	Causes Find to print only the count of lines that contained a match in each of the files.
/n	Causes each line to be preceded by its relative line number in the file.

The string must be enclosed by quotation marks.

Examples:

```
find "Fool's Paradise" book1.txt book2.txt
```

This command displays all lines from BOOK1.TXT and BOOK2.TXT (in that order) that contain the string "Fool's Paradise." The command

```
dir b: | find /v "date"
```

causes MS-DOS to display all names of the files on the disk in drive B that do not contain the string "date". Type double quotes around a string that already has quotes in it.

MESSAGES

Incorrect DOS version
: Find will only run on versions of MS-DOS that are 2.0 or higher.

Find: Invalid number of parameters
: You did not specify a string when issuing the Find command.

Find: Syntax error
: You typed an illegal string when issuing the Find command. Remember that the string must be enclosed by quotation marks.

Find: File not found <filename>
: The filename you have specified does not exist or Find cannot find it.

Find: Read error in <filename>
: An error occurred when Find tried to read the file specified in the command.

Find: Invalid parameter <option-name>
: You specified an option that does not exist.

NAME

Format

PURPOSE

Formats the disk in the specified drive to accept MS-DOS files.

SYNTAX

```
format <drive:>[/1][/4][/8][/b][/n:<xx>]
[/t:<yy>] [/v][/s]
```

COMMENTS

This command initializes the directory and file allocation tables on a disk. You must use this command to format all new disks before MS-DOS can use them.

You must specify the drive that you want to format.

Format uses the drive type to determine the default format for a disk.

When you format a hard disk, Format prompts you with the following:

```
Enter current Volume Label for drive (x:)
```

Press the Return key if your hard disk does not have a volume label. (Note: If your hard disk has never been formatted before, or if it has a bad boot sector, Format will not prompt you for a volume label.)

If the volume label that you enter does not match the label on the hard disk, Format displays the message:

```
Invalid Volume ID Format failure
```

Otherwise Format continues:

```
WARNING, ALL DATA ON NON-REMOVABLE DISK
DRIVE x: WILL BE LOST! Proceed with Format
(Y/N)?_
```

If you want to format your hard disk, type "Y" and press the Return key. If you do not want to format your hard disk, type "N" and press Return.

The /1 switch formats a disk for single-sided use, even if the disk or drive is double-sided. If the drive is double-sided and you do not specify this switch, you will not be able to use it in a single-sided drive.

The /4 switch formats a double-sided disk in a high-capacity disk drive. You should note that you may not be able to read disks formatted with this switch reliably in a single- or double-sided drive.

The /8 switch formats a disk for 8 sectors per track. If you do not specify this switch, Format defaults to either 9 or 15 sectors per track (depending on the type of drive being used). You should note that Format always creates either 9 or 15 sectors per track, but it tells MS-DOS to use only 8 sectors per track if you specify this switch.

The /b switch formats a disk with 8 sectors per track and allocates space for the hidden system files. If you this switch, you can place any version of MS-DOS on the disk by using that version's Sys command. If you do not use the /b switch, you can only place MS-DOS version 3.2 on the disk with the Sys command. You cannot use the /s or the /v switches with the /b switch.

The /n:<xx> option specifies the number of sectors per track that Format uses to format a floppy disk.

The /t:<yy> option specifies the number of tracks that Format places on a floppy disk.

The /v switch causes Format to prompt for a volume label after the disk is formatted. A volume label can be up to 11 characters long and is used to identify the disk. An example of a volume label is PROGRAMS.

If the /s switch is specified, it must be the last switch typed; then Format copies operating system files from the disk in the default drive to the newly formatted disk. The files are copied in the following order:

```
IO.SYS
MSDOS.SYS
COMMAND.COM
```

If the operating system isn't on the default drive, Format prompts you to insert a system disk in the default drive (or in drive A if the default drive is a non-removable drive).

When formatting is complete, Format displays a message showing the total disk space, any space marked as defective, the total space used by the operating system (when you use the /s switch), and the space available for your files.

The following table shows which switches you can use for certain types of disks:

Disk Type	Valid Switches
160/180KB	/1 /4 /8 /b /n /t /v /s
320/360KB	/1 /4 /8 /b /n /t /v /s
720KB	/n /t /v /s
1.2KB	/n /t /v /s
hard disk	/v /s

Notes:

Formatting destroys any previously existing data on a disk.

Format ignores drive assignments created with the Assign command.

You should not use Format with drives used in the Assign, Join, or Subst commands.

You cannot Format drives over the network.
Format returns the following errorlevel codes:

0 =	Successful completion
3 =	Terminated by user (Control-C)
4 =	Fatal error (any error other than 0, 3, or 5)
5 =	"N" response to hard disk prompt, "Proceed with format (Y/N)?"

You can check these exit codes by using the errorlevel option with the If batch processing command.

MESSAGES

Attempted write-protect violation
: The disk you are trying to format is write protected.

Bad Partition Table
: This message means that there is no DOS partition on the hard disk. You must run Fdisk to create a DOS partition on your hard disk.

Cannot format an ASSIGNed or SUBSTed drive
: You attempted to format a drive which is actually mapped to another drive by the Assign or Subst command. Run Assign or Subst again and clear all drive assignments.

Cannot FORMAT a Network drive
: You cannot format drives that are redirected over the Network.

Disk unsuitable for system disk
: The Format program detected a bad track on the disk where system files should reside. You should use this disk to store data only.

Drive letter must be specified
: You must specify the name of the drive that you want to Format.

Enter current Volume Label for drive (x:)
: Format asks you to enter the current volume label for verification before it formats the hard disk in the specified drive.

Error reading/writing partition table
: Format could not read or write the partition table. You should run Fdisk on the disk and then try formatting it again.

Format another (Y/N)?
: Format displays this message when it has finished formatting the disk in the specified drive and is ready to format another disk with the same format. Press "Y" to continue, and "N" to quit formatting. If you accidentally type Y, you can abort the format process by typing Control-C in response to the "Strike any key to begin formatting" message.

Format complete
: Format displays this message when it has finished formatting the disk in the specified drive.

Format failure
: MS-DOS could not format the disk. This message is usually displayed with an explanation as to why MS-DOS could not format the disk.

Format not supported on drive (x:)
: You cannot use Format to format this drive.

Head: (hh) Cylinder: (cc)
: Format displays the current head and cylinder number before formatting it.

Incorrect DOS version
: You cannot use Format with an earlier version of MS-DOS.

Insert diskette for drive (x:)
and strike ENTER when ready
: You should insert a disk in the appropriate drive and press the Return key to begin formatting.

Insert DOS disk in drive (x:)
: Format displays this message if you specified the /s switch to copy the MS-DOS system files, but your default disk does not contain them.

Insert new diskette for drive (x:)
and strike ENTER when ready
: This message appears when you are using a single drive to format a disk. You should insert a disk in the appropriate drive and press the Return key to begin formatting. If there is any data on the disk, Format will destroy it. Press Control-C if you don't want to format the disk.

Insufficient memory for system transfer
: Your memory configuration is insufficient to transfer the MS-DOS system files IO.SYS and MS-DOS.SYS with the /s switch.

Invalid characters in volume label
: The volume label should contain only up to 11 alphanumeric characters.

Invalid device parameters from device driver
Format displays this message when the number of hidden sectors is not evenly divisible by the number of sectors per track (i.e., the partition does not start on a track boundary). This might happen if you tried to format a hard disk that previously had been formatted with MS-DOS 2.x without first running Fdisk.

Invalid drive specification
You must specify a valid drive.

Invalid parameter
One of the switches that you have specified is wrong.

Invalid Volume ID
Format displays this message if you enter a volume label that doesn't match the label on the hard disk you want to format. It then quits the format process.

Parameters not compatible
You have specified switches that are mutually exclusive.

Parameters not compatible with fixed disk
You have used a switch that is not compatible with the specified drive.

Parameters not supported
You have specified parameters that MS-DOS does not support.

Parameters not supported by Drive
Format displays this message when the device driver for this drive does not support Generic IOCTL function requests.

Re-insert diskette for drive (x:)
Format displays this message after it has read in system files from another disk.

System transferred
The system files MS-DOS.SYS and IO.SYS have been transferred during formatting.

Track 0 bad - disk unusable

Format can accommodate for defective sectors on the disk except for those near the beginning. In this case, use another disk. This message can also occur when you are formatting high density media either with the Format /4 switch or in a standard density drive.

Volume label (11 characters, ENTER for none)?

This message appears in response to the Format /v switch. Specify a volume label or press the Return key to indicate that you do not want a volume label for this disk.

WARNING, ALL DATA ON NON-REMOVABLE DISK
DRIVE (x:) WILL BE LOST!
Proceed with Format (Y/N)?

There is data on the hard disk that you are trying to format. If you want to lose the data and format the disk, press Y (for "Yes"). If you do not want the files on your hard disk erased, press N (for "No"). Copy the files onto floppy disks and repeat the Format command.

E

NAME

Graftabl

PURPOSE

Loads additional character data into a table in memory for use with a color or graphics adapter.

SYNTAX

```
graftabl
```

COMMENTS

Graftabl loads a table of the ASCII characters 128 through 255 into memory. If you have a color or graphics adapter, this table lets you display foreign language characters when you are in graphics mode.

After Graftabl loads the character table, it displays the message:

```
GRAPHICS CHARACTERS LOADED
```

You can load the graphics table only once each time you start MS-DOS. If you try to load the graphics table a second time, Graftabl displays the following message:

```
GRAPHICS CHARACTERS ALREADY LOADED
```

Note

This command increases the size of MS-DOS resident in memory.

E

NAME

Graphics

PURPOSE

Lets you print a graphics display screen on a printer when you are using a color or graphics monitor adapter.

SYNTAX

```
graphics [<printer>] [/r] [/b]
```

COMMENTS

The printer option may be one of the following:

COLOR1	Prints on an IBM Personal Computer Color Printer with black ribbon.
COLOR4	Prints on an IBM Personal Computer Color Printer with RGB (red, green, blue, and black) ribbon.
COLOR8	Prints on an IBM Personal Computer Color Printer with CMY (cyan, magenta, yellow, and black) ribbon.
COMPACT	Prints on an IBM Personal Computer Compact Printer.
GRAPHICS	Prints on an IBM Personal Graphics Printer.

If you do not specify the printer option, Graphics defaults to the GRAPHICS printer type.

The /r switch prints black and white (as seen on the monitor) on the printer. The default is to print black as white and white as black.

The /b switch prints the background in color. This option is valid for COLOR4 and COLOR8 printers.

To print the screen, press the Shift and Printscreen keys at the same time. If the computer is in 320x200 color graphics mode, and if the printer type is COLOR1 or GRAPHICS, Graphics prints the screen contents with up to four shades of gray. If the computer is in 640x200 color graphics mode, Graphics prints the screen contents on the paper sideways.

Note

This command increases the size of MS-DOS resident in memory.

E

NAME

Join

PURPOSE

Joins a disk drive to a specific path.

SYNTAX

```
join <drive:> <drive:><pathname> [/d]
```

COMMENTS

If the path does not exist, MS-DOS tries to make a directory with that path. The path must be empty. After you issue the Join command, the first drive name becomes invalid, and if you try to use it, MS-DOS displays the error message, "Invalid drive."

The Join command removes the distinction that physical drives are separately addressable by drive letter. This way you can always refer to all the directories on a specific drive by one pathname. If the path already existed before the Join command was typed, it will not be usable while the join is in effect.

You can join a drive only at the root. The following command will work:

```
join d: c:\memos
```

The following command will NOT work:

```
join d: c:\memos\june
```

To deassign a splice ("unjoin"), use the following format:

```
join <drive:> /d
```

where <drive:> represents the source drive and the /d switch turns off the Join command. You can specify this switch immediately after the drive name.

If you just type:

```
join
```

MS-DOS displays the current drives that are joined.

MESSAGES

Directory not empty
: You can only join onto a directory that is empty.

Incorrect number of parameters
: You specified too many or too few options in the command line.

Not enough memory
: There is not enough memory for MS-DOS to run the command.

NAME

Label

PURPOSE

Creates changes, or deletes the volume identification label on a disk.

SYNTAX

```
label [<drive:>] [<volume label>]
```

COMMENTS

The volume label can be up to 11 characters long. The Label command truncates the volume label if you enter more than 11 characters.

You should not use the following characters in a volume label:

```
* ?  / \ | .  , ;  :  + = < > [ ]
```

The volume label may include spaces, but may not include tabs.

If you do not specify a volume label, Label prompts:

```
Volume in drive X is xxxxxxxxxxx
Volume label (11 characters, ENTER for
none)? _
```

If the disk does not already have a volume label, Label prompts:

```
Volume in drive X has no label
Volume label (11 characters, ENTER for
none)? _
```

Type the volume label that you want and press the Return key.

If you want to delete the volume label, just press the Return key. Label prompts with the message:

```
Delete current volume label (Y/N)?_
```

If you press Y, Label deletes the volume label on the disk. Otherwise, the volume label remains unchanged.

MESSAGES

Invalid characters in volume label
Volume label (11 characters, ENTER for none)?

You cannot use the characters:

```
* ? / \ | . , ; : + = < > [ ]
```

or the tab character in a volume label. Reenter the volume label.

Cannot LABEL a SUBSTed or ASSIGNed drive

You cannot label a drive that has been assigned via the ASSIGN command or that has been substituted with a string via the SUBST command.

Cannot label a network drive

You cannot label a drive that is shared on a network server station.

Invalid drive specification

You cannot label a drive that is not logically assigned.

NAME

Mkdir

SYNONYM

MD

PURPOSE

Makes a new directory.

SYNTAX

```
mkdir [<drive:>]<pathname>
```

COMMENTS

This command creates a multilevel directory structure. When you are in your root directory, you can create subdirectories by using the Mkdir command. The command:

```
mkdir \user
```

creates a subdirectory \USER in your root directory. To create a directory named JOE under \USER, type:

```
mkdir \user\joe
```

When you create directories with Mkdir, they always appear under your working directory unless you explicitly specify a different pathname with Mkdir.

MESSAGES

Unable to create directory

MS-DOS could not create the directory you specified. Check to see that there is not a name conflict. You may have a file by the same name, or the disk may be full.

E

NAME

Mode

PURPOSE

Sets operation modes for devices.

SYNTAX

Parallel printer mode:
```
mode LPT<n>:[<chars>][,[<lines>][,P]]
```

Asynchronous communications mode:
```
mode COM<m>:<baud>[,<parity>[,<databits>
[,<stopbits>[,P]]]]
```

Redirecting parallel printer output:
```
mode LPT<n>:=COM<m>:
```

Display modes
```
mode <display>
mode [<display>],<shift>[,T]
```

COMMENTS

For parallel printer modes:

n	Specifies the parallel printer port number: 1, 2, or 3.
chars	Specifies either 80 or 132 characters per line.
lines	Specifies either 6 or 8 lines per inch, vertical spacing.
P	Specifies that Mode will continuously try to send output to the printer if a time-out error occurs.

The default settings are LPT1, with 80 characters per line and 6 lines per inch. You can break out of a time-out loop by pressing Control-Break.

For asynchronous communication modes:

m	Specifies the asynchronous communications (COM) port number: 1 or 2.

baud	Specifies the transmission rate: 110, 150, 300, 600, 1200, 2400, 4800, or 9600. You need to specify at least the first two digits of each number.
parity	Specifies the parity, either N (none), O (odd), or E (even).
databits	Specifies either 7 or 8 bits of data.
stopbits	Specifies the number of stopbits, either 1 or 2.
P	Specifies that Mode is using the COM port for a serial printer, and will continuously retry if time-out errors occur.

The default settings are COM1 with even parity and 7 databits. If <baud> is 110, then the default number of stop bits is 2, otherwise, the default is one stop bit.

For redirecting parallel printer output (to an asynchronous communications port):

n	Specifies the parallel printer port number: 1, 2, or 3.
m	Specifies the asynchronous communications port number: 1 or 2.

Note: MS-DOS does not check to see if you have redirected more than one parallel port to a single COM port.

For setting display modes:

display	Specifies one of: 40, 80, BW40, BW80, CO40, CO80, or MONO, where: 40 indicates 40 characters per line. 80 indicates 80 characters per line. BW and CO refer to a color graphics monitor adapter with (BW), or with color enabled (CO). color disabled MONO specifies a monochrome display adapter that always has a display width of 80 characters.

shift	Specifies the direction that you want to shift the display, either R (right), or L (left).
T	Specifies a test pattern that you can use to align the display.

If you specify the T option, Mode asks if the screen is aligned properly. If you type N, Mode repeats the shift and asks if the screen is aligned properly. The command ends when you type Y.

MESSAGES

COM port does not exist
: You have specified an invalid COM port.

Do you see the leftmost 0? (Y/N)
: Mode displays this message to help you align the test pattern on your screen.

Do you see the rightmost 9? (Y/N)
: Mode displays this message to help you align the test pattern on your screen.

Illegal device name
: Your computer does not recognize this device name.

Infinite retry on parallel printer timeout
: Your printer is probably off-line or not ready. If the printer appears to be ready, you should press the Control-Alt-Del keys to reset the computer.

Invalid baud rate specified
: You have specified an incorrect baud rate. Valid choices are 110, 150, 300, 600, 1200, 2400, 4800, and 9600. You must specify at least the first two digits of the baud rate.

LPT#: not redirected
: The status of LPT# is UN-REDIRECTED. Check to see that you have specified the proper options if you wanted to redirect this LPT device to a COM device.

LPT#: redirected to COM#:
: Output on the parallel printer port will now be sent to this asynchronous communications port.

LPT#: set for 132
The parallel printer port has been set for 132 columns.

LPT#: set for 80
The parallel printer port has been set for 80 columns.

No COM: ports
Your computer does not have a COM: port.

Printer lines per inch set
Mode has set the number of lines per inch for the printer.

Resident portion of MODE loaded
Part of the Mode program is now resident in memory.

Unable to shift Screen
Mode is unable to shift the test pattern on the screen any farther.

E

NAME

More

PURPOSE

Sends output to the console one screen at a time.

SYNTAX

```
more
```

COMMENTS

More is a filter that reads from standard input (such as a command from your terminal) and displays one screen of information at a time. The More command then pauses and displays the --More-- message at the bottom of your screen.

Pressing the Return key displays another screen of information. This process continues until all the data has been read.

The More command is useful for viewing a long file one screen at a time. If you type:

```
type myfiles.new | more
```

MS-DOS displays the file MYFILES.NEW one screen at a time.

I

NAME

Path

PURPOSE

Sets a command search path.

SYNTAX

```
path [[<drive:>][<pathname>];
[<drive:>][<pathname>]...]
```

COMMENTS

This command allows you to tell MS-DOS which directories should be searched for external commands after MS-DOS searches your working directory. The default value is no path.

To tell MS-DOS to search the \USER\JOE directory for external commands, type:

```
path \user\joe
```

MS-DOS now searches the \USER\JOE directory for external commands until you set another path or exit MS-DOS.

You can tell MS-DOS to search more than one path by specifying several pathnames separated by semicolons. For example:

```
path \user\joe;b:\user\sue;\bin\dev
```

tells MS-DOS to search the directories specified by the above pathnames to find external commands. MS-DOS searches the pathnames in the order specified in the Path command.

The command, Path, with no options prints the current path. If you specify:

```
path ;
```

MS-DOS sets the NUL path. This means that only the working directory is searched for external commands.

MESSAGES

No path

You typed Path with no options but have not set a command search path.

E

NAME

Print

PURPOSE

Prints text files on a lineprinter while you are processing other commands (usually called "background printing").

SYNTAX

```
print [<drive:>][<pathname>] [/d:<device>]
[/b:<size>] [/q:<value>] [/t] [/c] [/p]
```

COMMENTS

Use the Print command only if you have a lineprinter attached to your computer. The following switches are provided with this command:

/d	DEVICE. Specifies the print device. If not specified, the default device is PRN. Print prompts for a print device.

The following switches are allowed only the first time you run the print command after starting MS-DOS.

/b	BYTES. This switch sets the size in bytes of the internal buffer. Increasing the value of /b speeds up the Print command.
/q	QUEUE. Specifies the number of files allowed in the print queue if you want more than 10. The minimum value for the /q switch is 4; the maximum is 32.
/t	TERMINATE. This switch deletes all files in the print queue (those waiting to be printed). A message to this effect will be printed.
/c	CANCEL. This switch turns on cancel mode. The preceding filename and all following filenames are removed from the print queue.
/p	PRINT. This switch turns on print mode. The preceding filename and all following filenames are then added to the print queue.

If you use Print without options, it displays the contents of the print queue on your screen without affecting the queue.

Examples:

```
print /t
```

Empties the print queue.

```
print a:templ.tst /c a:temp2.tst a:temp3.tst
```

Removes the three files indicated from the print queue.

```
print templ.tst /c temp2.tst /p temp3.tst
```

Removes TEMP1.TST from the queue, and adds TEMP2.TST and TEMP3.TST to the queue.

Note

Each print queue entry may contain a maximum of 64 characters, including the drive name. Therefore, you may need to change directories if you want to print files in deep subdirectories.

MESSAGES

All files canceled

If you specify the /t ("terminate") switch, MS-DOS prints "All files canceled by operator" on your printer. If the current file being printed is canceled by a /c, the message "File canceled by operator" is printed.

Cannot open (filename)

Either MS-DOS cannot find the specified file to print or the file does not exist. Check the command for a valid filename.

Errors on list device indicate that it may be offline. Please check it

Your printer is turned off.

(filename) file not found
: You switched disks when a file was queued up, but before it started to print. Reissue the Print command for that file.

List output is not assigned to a device
: This message is displayed if the "Name of list device" specified to the above prompt is invalid. Subsequent attempts return the same message until you specify a valid device.

Name of list device [PRN:]
: This prompt appears when Print is run the first time and the /d switch is not specified. Any current device may be specified and that device then becomes the Print output device. As shown in the brackets, you can press the Return key so that MS-DOS uses the default list device (PRN).

No files match drive:xxxxxxxx.xxx
: A drive and filename were given for files to add to the queue, but no files match.

NOTE: If there are no files in the queue to match the canceled filename, no error message appears.

Print queue is empty
: There are no files in the print queue.

Print queue is full
: There is room for 10 files in the queue. If you attempt to put more than 10 files in the queue, this message appears on the screen. To add more files to the queue, refer to the /q switch in the list above.

Resident part of Print installed
: This is the first message that MS-DOS displays when you issue the Print command. It means that available memory has been reduced by several thousand bytes to process the Print command along with other processes.

I

NAME

Prompt

PURPOSE

Changes the MS-DOS command prompt.

SYNTAX

```
prompt [<text>]
```

COMMENTS

This command allows you to change the MS-DOS system prompt (for example, A>). If no text is typed, the prompt is set to the default prompt, which is the default drive designation. You can set the prompt to a special prompt, such as the current time, by using the characters indicated below.

You can use the following characters in the prompt command to specify special prompts. They must all be preceded by a dollar sign ($):

Specify This Character	**To Get This Prompt:**
$	The '$' character
t	The current time
d	The current date
p	The working directory of the default drive
v	The version number
n	The default drive
g	The '>' character
l	The '<' character
b	The '|' character
_	A Return-line feed sequence
s	A space (leading only)
e	ASCII code X'1B' (escape)

Examples:

```
     Prompt $p
```

Sets the drive prompt to <drive:><current directory>.

```
     Prompt Time = $t$_Date = $d
```

Sets a two-line prompt which prints:

```
     Time = (current time)
     Date = (current date)
```

If your terminal has an ANSI escape sequence driver, you can use escape sequences in your prompts. For example:

```
     Prompt  $e[7m$n:$e[m
```

Sets the prompts in inverse video mode and returns to video mode for other text.

NAME

Recover

PURPOSE

Recovers a file or an entire disk containing bad sectors.

SYNTAX

```
recover [<drive:>]

  or

recover <drive:>[<pathname>]
```

COMMENTS

If a sector on a disk is bad, you can recover either the file containing that sector (without the bad sector) or the entire disk (if the bad sector was in the directory).

To recover a particular file, type:

```
recover <filename>
```

This causes MS-DOS to read the file sector by sector and to skip the bad sector(s). When MS-DOS finds the bad sector(s), the sector(s) are marked and MS-DOS will no longer allocate your data to that sector.

To recover a disk, type

```
recover <drive:>
```

where <drive:> is the letter of the drive containing the disk to be recovered.

MESSAGES

File not found

MS-DOS cannot find the file that you specified. Check to see that the pathname is accurate and that the file exists in the directory you specified.

(xxxx) of (xxxx) bytes recovered

This message tells you the number of bytes that MS-DOS was able to recover from the disk.

Warning - directory full
: The root directory is too full for Recover processing. Delete some files in the root directory to free space.

I

NAME

Ren (Rename)

SYNONYM

Rename

PURPOSE

Changes the name of a file.

SYNTAX

```
ren [<drive:>]<pathname> <pathname>
```

COMMENTS

You must supply a drive name if the file resides on other than the default drive. Any drive name for the third option (pathname) is ignored. You cannot rename files across drives.

Wildcards may be used in either option. All files matching the first filename are renamed. If wildcards appear in the second filename, corresponding character positions will not be changed. For example, the following command changes the extension of all filenames ending in .LST to .PRN:

```
ren *.lst *.prn
```

In the next example, Ren renames the file CHAP10 on drive B to PART10:

```
ren b:chap10 part10
```

The file remains on drive B.

MESSAGES

File not found

You tried to rename a file to a name already present in the directory.

NAME

Replace

PURPOSE

Updates previous versions of files.

SYNTAX

```
replace [<drive:>]<pathname> [<drive:>][<path>]
[/a][/d][/p][/r][/s][/w]
```

COMMENTS

The Replace command lets you easily update files on your hard disk with new versions of software.

Replace performs two functions:

1. By default, it replaces files in the target directory with files in the source directory that have the same name. You may use wildcards in source filenames.

2. When you specify the /a switch, Replace adds files that exist in the source directory (but NOT in the target directory) to the target directory.

The /a switch adds new files to the target directory instead of replacing existing ones. You may not use this switch with either the /d or /s switches.

The /d switch replaces files in the target directory only if the source files are newer than the corresponding target files. This switch is incompatible with the /a switch.

The /p switch prompts you before replacing a target file or adding a source file.

```
Replace (filename) ? (Y/N) _
```

The /r switch replaces read-only files as well as unprotected files. If you do not specify this switch, any attempt to replace a read-only file causes an error and stops the replace process.

The /s switch causes Replace to search all subdirectories of the target directory while it replaces matching files. This switch is incompatible with the /a switch. Replace never searches subdirectories in the source path.

The /w switch waits for the user to hit any key before replacing files. If you do not specify this switch, Replace begins replacing or adding files immediately.

As files are replaced or added, Replace displays the filenames on the screen; then at the conclusion of the replace operation, it displays a summary line:

```
NNN file(s) added/replaced
          or
No files added/replaced
```

You cannot use the Replace command to update hidden files or system files.

EXAMPLES

Replacing Files:

Suppose your hard disk, drive C, contains several files of client names and phone numbers. To replace these files with the latest version of this file that exists on the disk in drive A, you would type:

```
replace a:\phones.cli c:\ /s
```

This command replaces every file on drive C that is named PHONES.CLI with the file PHONES.CLI from the root directory on drive A.

Adding Files:

Suppose you want to add some new printer device drivers to a directory called C:\MSTOOLS, which already contains several printer driver files for a word processor. To do this, you would type:

```
replace a:*.prd c:\mstools /a
```

This command adds any files from the default directory of drive A with an extension of PRD (that do not currently exist in the \MSTOOLS directory on drive C) to C:\MSTOOLS.

If Replace encounters an error, it returns one of the following errorlevel codes:

1	Command line error
2	File Not Found
3	Path Not found
5	Access Denied
8	Insufficient Memory
15	Invalid Drive
Other	Standard MS-DOS error

You can test for these codes by using the errorlevel option with the batch processing If command.

MESSAGES

Access denied (filename)
: You tried to replace a write-protected or locked file.

File cannot be copied onto itself (filename)
: You attempted to copy a file to itself.

Insufficient disk space
: There is not enough room on the target disk for the replace operation.

Insufficient memory
: You do not have enough memory available to use the Replace command.

Invalid drive specification (x:)
: Drive X does not exist.

Invalid parameter
: You used an invalid parameter or switch.

No files added/replaced
: The Replace command did not add or replace any files.

No files found (filename)
: Replace could not find matching source or target files.

Parameters not compatible
: You have specified switches that are incompatible with each other.

Path not Found (path)
: The source or target path did not exist.

Path too long
: The path name you specified was too long. You may have to change directories to replace files in deep subdirectories.

Press any key to begin adding files
: When you specify the /w switch, Replace displays this message to prompt you to start replacing files.

Source path required
: You did not specify a source path for Replace.

Unexpected DOS Error (NNN)
: An unexpected error occurred, where (NNN) is the MS-DOS error number.

E

NAME

Restore

PURPOSE

Restores one or more files that were backed up using either the Microsoft or the IBM Backup command.

SYNTAX

```
restore <drive:> [<drive:>][<pathname>][/s][/p]
[/a:<date>] [/b:<date>] [/e:<time>] [/L:<time>]
[/m] [/n]
```

COMMENTS

The Restore command can restore files from disks of different media. For example, you can restore files from:

Hard disk to floppy disk
Floppy disk to floppy disk
Floppy disk to hard disk
Hard disk to hard disk

The first option you specify is the drive name for the disk containing the backed up files. The second and third options are the drive and pathname of the files you want to restore.

This restore program and the one supplied by IBM are compatible except for the /b, /a, /e, /L, /m, and /n switches. You can use the following switches with the MS-DOS Restore command:

/s	Restore subdirectories also.
/p	If any hidden or read-only files match the file specification, prompt for permission to restore them.
/b	Restore only those files that were last modified on or before the given date.
/a	Restore only those files that were last modified on or after the given date.
/e	Restore only those files that were last modified at or earlier than the given time.

/L	Restore only those files that were last modified at or later than the given time.
/m	Restore only those files that have been modified since the last backup.
/n	Restore only those files that no longer exist on the destination disk.

If you are restoring files that were backed up using a previous version of MS-DOS, you should use the /p switch. This switch prompts you before restoring the MS-DOS system files. You should type "N" when Restore asks if you want to restore these files to avoid destroying the newer version of MS-DOS on your disk.

The Restore program returns the following errorlevel codes:

0	Normal completion
1	No files were found to restore
2	Some files not restored due to sharing conflicts
3	Terminated by user
4	Terminated due to error

You can use the batch processing If command to perform error processing based on the errorlevel returned by Restore.

I

NAME

Rmdir (Remove Directory)

SYNONYM

rd

PURPOSE

Removes a directory from a multilevel directory structure.

SYNTAX

```
rmdir <pathname>
```

COMMENTS

This command removes a directory <u>that is empty</u> except for the . and .. shorthand symbols. You must delete all files in the directory first.

Example:

To remove the \USER\JOE directory, first issue a Dir command for that path to ensure that the directory is empty. Then type:

```
rmdir \user\joe
```

The directory will be deleted from the directory structure.

NAME

Set

PURPOSE

Sets one string value in the environment equal to another string for use in later programs.

SYNTAX

```
set [<string=string>]
```

COMMENTS

This command is meaningful only if you want to set values that will be used by programs you have written.

When MS-DOS sees a Set command, it inserts the entire string into a part of memory reserved for "environment" strings. If the name already exists in the environment, it is replaced by the new string. If you type the Set command with only the first string, the associated string name is removed from the environment. If you type Set with no options, MS-DOS displays the current environment settings.

An application program can get a listing of all environment values that have been set by examining its environment. (Environments are passed in the Program Segment Prefix). Refer to Chapter 4, "MS-DOS Control Blocks and Work Areas," in the MS-DOS Programmer's Reference Manual for more information.

The Set command can also be used in batch processing. In this way, you can define your replaceable parameters with names instead of numbers. If your batch file contains the statement "LINK %FILE%", you can set the name that MS-DOS will use for that variable with the Set command. The command

```
set file=domore
```

replaces the %FILE% parameter with the filename DOMORE. Therefore, you do not need to edit each batch file to change the replaceable parameter names. Note that when you use text (instead of numbers) as replaceable parameters, the name must be ended by a percent sign.

Examples:

The command

```
set TTY=VT52
```

sets your TTY value to VT52 until you change it with another Set command. If you just type:

```
set
```

MS-DOS displays the current environment strings.

E

NAME

Share

PURPOSE

Installs file sharing and locking.

SYNTAX

```
share [/f:<space>][/L:<locks>]
```

COMMENTS

The Share command is only used when networking is active. It is included in the AUTOEXEC.BAT file to install shared files. Refer to the Microsoft Networks Manager's Guide to learn about shared files.

Use the /f:<space> switch to allocate file space (in bytes) for the area MS-DOS uses to record filesharing information. Each file that is open needs the length of the full filename plus 11 bytes (the average pathname is 20 bytes). The default value for the /f switch is 2048 bytes.

The L:<locks> switch allocates the number of locks you want to allow. The default value for the /L switch is 20 locks.

Once you have used the Share command in an MS-DOS session, all read and write rquests are checked by MS-DOS.

Example:

The following example loads file sharing and uses the default values for the /f and /L switches:

```
share
```

MESSAGES

Incorrect parameter
    One of the options you specified is wrong.

Not enough memory
    There is not enough memory for MS-DOS to run the command.

Share Already Installed
    You can install Share only once.

NAME

Sort

PURPOSE

Sort reads standard input, sorts the data, then writes it to your terminal screen, a file, or to another device.

SYNTAX

```
sort [<drive:>] [<pathname>] [/r] [/+<n>]
```

COMMENTS

Sort can be used, for example, to alphabetize a file by a certain column. There are two switches which allow you to select options:

/r	Reverse the sort; that is, sort from Z to A.
/+<n>	Sort starting with column n where n is some number. If you do not specify this switch, Sort will begin sorting from column 1.

Examples:

This command reads the file UNSORT.TXT, reverses the sort, and then writes the output to a file named SORT.TXT:

```
sort /r <unsort.txt> sort.txt
```

The following command pipes the output of the directory command to the Sort filter. The Sort filter sorts the directory listing starting with column 14 (this is the column in the directory listing that contains the file size), then sends the output to the screen. The result of this command is a directory sorted by file size:

```
dir | sort /+14
```

The command:

```
dir | sort /+14 | more
```

does the same thing as the command in the previous example, except that the More filter gives you a chance to read the sorted directory one screen at a time.

NAME

Subst

PURPOSE

Substitutes a string alias for a pathname.

SYNTAX

```
subst [<drive:>] [<pathname>][/d]
```

COMMENTS

The Subst command creates or deletes a "virtual drive" by associating a pathname with a drive letter.

When MS-DOS sees a drive that was created with the Subst command, it replaces the reference with the new pathname.

If you type

```
subst
```

MS-DOS displays the names of the "virtual drives" in effect.

Use the /d switch to delete an associated drive or pathname. You can specify this switch immediately after the drive name.

Example:

The command:

```
subst z:  b:\usr\fred\forms
```

creates a virtual drive z for the pathname "b:\usr\fred\forms." Now, instead of typing the full pathname, you can get to this directory by simply typing z:.

MESSAGES

Incorrect number of parameters
  You specified too many or too few options in the command.

Not enough memory
  There is not enough memory for MS-DOS to run the command.

NAME

Sys (System)

PURPOSE

Transfers the MS-DOS system files from the disk in the default drive to the disk in the specified drive.

SYNTAX

```
sys <drive:>
```

COMMENTS

Sys is normally used to update the system or to place the system on a formatted disk that contains no files. You must type a drive letter with this command.

If IO.SYS and MSDOS.SYS are on the target disk, they must take up the same amount of space on the disk as the new system will need. This means that you cannot transfer system files from an MS-DOS 2.0 disk to an MS-DOS 1.1 disk. You must reformat the MS-DOS 1.1 disk with the MS-DOS Format command before the Sys command will work. The target disk must be completely blank or already have the system files IO.SYS and MSDOS.SYS. The transferred files are copied in the following order:

```
IO.SYS
MSDOS.SYS
```

IO.SYS and MSDOS.SYS are both hidden files that do not appear when you type the Dir command. COMMAND.COM (the command processor) is <u>not</u> transferred. Use the Copy command to transfer COMMAND.COM.

MESSAGES

No room for system on destination disk

There is not enough room on the target disk for the IO.SYS and MSDOS.SYS files.

Incompatible system size

The system files IO.SYS and MSDOS.SYS do not take up the same amount of space on the target disk as the new system will need.

NAME

Time

PURPOSE

Displays and sets the time.

SYNTAX

```
time [<hours>:<minutes>]
```

COMMENTS

If you just type "time", the following message is displayed:

```
Current time is (hh):(mm):(ss).(cc)
Enter new time:_
```

Press the Return key if you do not want to change the time shown. If you want to change the time after you have started MS-DOS, (for example, to 8:20 a.m.), type

```
time 8:20
```

in response to the MS-DOS prompt. The new time must be entered using numbers only; letters are not allowed. The allowed options are:

```
<hours> = 00-24
<minutes> = 00-59
```

Separate the hour and minute entries by a colon. You do not have to type the <ss> (seconds) or <cc> (hundredths of seconds).

If you do not type a valid time, MS-DOS displays the message:

```
Invalid time
Enter new time:_
```

MS-DOS then waits for you to type a valid time.

Like the Date command, the Time Command format can be changed with the Country command in the CONFIG.SYS file. Refer to Appendix D, "How to Configure Your System," for more information.

E

NAME

Tree

PURPOSE

Displays the path (and optionally list the contents) of each directory and subdirectory on the given drive.

SYNTAX

```
tree [<drive:>] [/f]
```

COMMENTS

The Tree command lists the full path of each directory, along with the names of each of their subdirectories.

The <drive:> option specifies the drive that you want to want to use. If you do not specify this option, Tree uses the default drive.

The /f switch causes Tree to display the names of the files in each directory.

Example:
If you want to print a list of the directory and file names on the disk in drive B, you can use the command:

```
tree b:  /f > prn
```

MESSAGES

No subdirectories exist
: The disk in the drive you specified does not contain subdirectories.

Invalid path
: You specified an invalid path.

Invalid drive specification
: The drive name that you specified does not exist or is invalid.

Invalid parameter
: The /f switch is the only valid parameter for Tree.

Incorrect DOS version
: You are using the Tree command with the wrong version of MS-DOS. Insert the correct version of DOS and try the command again.

I

NAME

Type

PURPOSE

Displays the contents of a file on the screen.

SYNTAX

```
type [<drive:>]<filename>
```

COMMENTS

Use this command to view a file without modifying it. (Use Dir to find the name of a file and EDLIN to change the contents of a file.) Note that when you use Type to display a file with tabs in it, all tabs are expanded to 8 spaces wide. A display of binary files causes control characters (such as Control-Z) to be sent to your computer, including bells, form feeds, and escape sequences.

I

NAME

Ver

PURPOSE

Prints MS-DOS version number.

SYNTAX

```
ver
```

COMMENTS

If you want to know what version of MS-DOS you are using, type:

```
ver
```

The version number will be displayed on your screen.

I

NAME

Verify

PURPOSE

Turns the verify switch on or off when writing to disk.

SYNTAX

```
verify [ON]

  or

verify [OFF]
```

COMMENTS

This command has the same purpose as the /v switch in the Copy command. If you want to verify that all files are written correctly to disk, you can use the Verify command to tell MS-DOS to verify that your files are intact (no bad sectors, for example). MS-DOS will perform a Verify each time you write data to a disk. You will receive an error message only if MS-DOS was unable to successfully write your data to disk.

Verify ON remains in effect until you change it in a program (by a Set Verify system call), or until you type "Verify OFF".

If you want to know what the current setting of Verify is, type:

```
verify
```

with no options.

NAME

Vol (Volume)

PURPOSE

Displays disk volume label, if it exists.

SYNTAX

```
vol [<drive:>]
```

COMMENTS

This command prints the volume label of the disk in a specific drive. If you do not type a drive letter, MS-DOS prints the volume label of the disk in the default drive.

MESSAGES

```
Volume in drive x has no label
```

The disk in the drive does not have a volume label.

E

NAME

Xcopy

PURPOSE

Copies files and directories, including lower level directories if they exist.

SYNTAX

```
xcopy [<drive:>][<path>]<filename>
[<drive:>][<path>][<filename>] [/a]
[/d:<mm-dd-yy>][/e][/m][/p][/s][/v][/w]
```

COMMENTS

The first drive, path, and filename parameters specify the file or directory that you want to copy. The second drive, path, and filename parameters specify the target. You must include at least one of the source parameters. If you omit the target parameters, Xcopy assumes you want to copy the files to the default directory.

If you do not specify the path option, Xcopy uses the default directory. The default filename is "*.*".

The /a switch copies source files that have their archive bit set. It does not modify the archive bit of the source file. Refer to the Attrib command for information on how to set the archive attribute.

The /d switch copies source files that you modified on or after the date specified by <mm-dd-yy>. Note that the date format may vary depending on the country code that you are using.

The /e switch copies any subdirectories, even if they are empty. You must use this switch with the /s switch.

The /m switch is similar to the /a switch since it copies archived files only; however, it turns off the archive bit in the source file. Refer to the Attrib command for information on how to set the archive attribute.

The /p switch prompts you with "(Y/N?)" allowing you to confirm whether you want to create each target file.

The /s switch copies directories and lower level subdirectories unless they are empty. If you omit this switch, Xcopy works within a single directory.

The /v switch causes Xcopy to verify each file as it is written to the target to make sure that the target files are identical to the source files.

The /w switch causes Xcopy to wait before it starts copying files. Xcopy displays the message:

```
Press any key when ready to start copying
files
```

You must press a key to continue, or press Control-C to abort the Xcopy command.

Example:

Because the Diskcopy command copies disks track-by-track, it requires your source and target disks to have the same format. So if you want to make a copy of a disk that contains files in subdirectories to a target disk that has a different format, you must use the Xcopy command. For example:

```
xcopy a: b: /s /e
```

copies all of the files and subdirectories (including any empty subdirectories) on the disk in drive A to the disk in drive B.

Note

The Xcopy command may prompt you to specify whether the target is a file or a directory. If you do not want to receive this prompt, type:

```
copy /b xcopy.exe mcopy.exe
```

This creates a new command called MCOPY.EXE. Now you can use the Mcopy command the same way you use the Xcopy command, but Mcopy automatically determines whether the target is a file or directory.

Mcopy uses the following rules for copying files:

1. If the source is a directory, the target is a directory.

2. If the source includes multiple files, the target is a directory.

3. If you append a backslash (\) to the end of the target name, the target is a directory. For example:

   ```
   xcopy payroll a:\workers\
   ```

   creates the directory A:\WORKERS if it doesn't already exist, and copies the file PAYROLL to it.

When correctly written programs exit back to DOS, they return an error code: 0 if no error occurred, or a value greater than zero if there was a problem. This error code can be tested in batch files, and it allows batch programmers to "branch" to an error-handling routine in the batch file.

If Xcopy encounters an error, it returns one of the following errorlevel codes:

0	Copy without error
1	No files found to copy
2	Control-C entered by user to terminate Xcopy
4	Initialization error There is not enough memory--invalid drive or command line syntax, file not found, or path not found.
5	Int 24 error occurred The user aborted from INT 24 error reading or writing disk.

You can test for these codes by using the errorlevel option with the batch processing If command.

MESSAGES

Access denied
: You tried to overwrite a read-only file.

Cannot perform a cyclic copy
: When you are using the /s switch, you may not specify a target that is a subdirectory of the source.

Cannot XCOPY to a reserved device
: You cannot copy files to a device.

Cannot XCOPY from a reserved device
: You cannot copy from a device to a file.

Does (name) specify a file name
or directory name on the target
(F = file D = directory)?
: Xcopy displays this prompt if the target directory does not exist.

File cannot be copied onto itself
: You have specified the same source and target.

File creation error
: Xcopy could not create the file. Run the Chkdsk program to determine if the disk is full.

Incorrect DOS version
: You must run the Xcopy command with the correct version of MS-DOS.

Invalid date
: You entered an illegal date. Refer to the Date command for information on how to enter the date.

Invalid drive specification
: You specified a drive name that does not exist.

Invalid parameter
: You specified an incorrect parameter. Check the syntax for this command and try again.

Invalid path
: The source path described a non-existent subdirectory.

Invalid number of parameters
: You specified too many or too few parameters. Check the syntax for this command and try again.

Path too long
: The path name you specified was too long. You may have to change directories to replace files in deep subdirectories.

Reading source file(s)...
: Xcopy is now reading the source files that you specified.

Unable to create directory
: MS-DOS cannot create a directory if a file with the same name already exists. Check to see that you typed the directory name correctly, or rename the file or directory if necessary.

3.4 BATCH PROCESSING COMMANDS

The following commands are called batch processing commands. They can add flexibility and power to your batch programs. The commands discussed are Echo, For, Goto, If, Pause and Shift.

If you are not writing batch programs, you do not need to read this section.

NAME

Echo

PURPOSE

Turns batch echo feature on and off.

SYNTAX

```
echo [ON]

  or

echo [OFF]

  or

echo [<message>]
```

COMMENTS

Normally, commands in a batch file are displayed ("echoed") on the screen when they are seen by MS-DOS. The command:

```
echo off
```

turns off this feature. Echo ON turns the echo back on.

If ON or OFF are not specified, the current setting is displayed.

Echo <message> is only useful if Echo is off and you are using a batch file. By typing Echo and a message in your batch file, you can print messages on the screen.

I

NAME

For

PURPOSE

Command extension used in batch and interactive file processing.

SYNTAX

```
for %%<c> in <set> do <command>
(for batch processing)

for %<c> in <set> do <command>
(for interactive processing)
```

COMMENTS

<c> can be any character except 0,1,2,3,..,9 to avoid confusion with the %0-%9 batch parameters.

```
     <set> is ( <item>* )
```

The %%<c> variable is set sequentially to each member of <set>, and then <command> is evaluated. If a member of <set> is an expression involving * and/or ?, then the variable is set to each matching pattern from disk. In this case, only one such item may be in the set, and any item besides the first is ignored.

Examples:

```
     for %%f in ( *.asm ) do masm %%f;

     for %%f in (report memo address) do del %%f
```

The first example binds the variable named %f to files ending with *.asm in the working directory. It then executes the command:

```
     masm <filename>
```

```
where <filename> could be:   bigfile.asm
                             sorter.asm
                             list.asm
```

The second example binds the variable %f to the files named report, memo, and address. It then deletes each of these files.

The '%%' is needed so that after batch parameter (%0-%9) processing is done, there is one '%' left. If only '%f' were there, the batch parameter processor would see the '%', look at 'f', decide that '%f' was an error (bad parameter reference) and throw out the '%f', so that the command For would never see it. If the For is not in a batch file, then only one '%' should be used.

MESSAGES

For cannot be nested
    Nesting of For statements is illegal.

I

NAME

Goto

PURPOSE

Command extension used in batch file processing.

SYNTAX

goto <label>

COMMENTS

Goto causes commands to be taken from the batch file beginning with the line after the <label> definition. If no label has been defined, the current batch file will terminate.

Example:

```
:one
rem first program
goto two

.
.
.

:two
rem second program
```

sends the program processor to the label named "two".

Starting any line in a batch file with a colon (:) causes the line to be ignored by batch processing. The characters following Goto define a label.

NAME

If

PURPOSE

Command extension used in batch file processing.

SYNTAX

```
if <condition> <command>
```

COMMENTS

The parameter <condition> is one of the following:

ERRORLEVEL <number>
True if, and only if, the previous program executed by COMMAND.COM had an exit code of <number> or higher. (An exit code is set by a specific program. It is returned by the operating system after the program has finished. Later program tasks may be performed based on the value of this number.)

<string1> == <string2>
True if, and only if, <string1> and <string2> are identical after parameter substitution. Strings may not have embedded separators.

EXIST <filename>
True if, and only if, <filename> exists.

NOT <condition>
True if, and only if, <condition> is false.

The If statement allows conditional execution of commands. When the <condition> is true, then the <command> is executed. Otherwise, the <command> is ignored.

Examples:

```
if not exist data1 echo can't find file

if not errorlevel 3 link $1,,;
```

NAME

Pause

PURPOSE

Suspends execution of the batch file.

SYNTAX

```
pause [<comment>]
```

COMMENTS

When a batch file is running, you may need to change disks or perform some other action. Pause suspends execution until you press any key, except Control-C.

When the command processor encounters Pause, it prints:

```
Strike a key when ready . . .
```

If you press Control-C, MS-DOS displays the following message:

```
Terminate batch job (Y/N)?
```

If you type Y in response to this prompt, the batch file ends and control returns to the operating system. Therefore, Pause can be used to break a batch file into pieces, allowing you to end the batch command file at any intermediate point.

The comment is optional and may be typed on the same line as Pause. You may also want to prompt the user of the batch file with some meaningful message when the batch file pauses. For example, you may want to change disks in one of the drives. An optional prompt message may be given in such cases. The comment prompt will be displayed before the "Strike a key" message.

NAME

Rem (Remark)

PURPOSE

Displays remarks which are on the same line as the Rem command in a batch file during execution of that batch file.

SYNTAX

```
rem [<comment>]
```

COMMENTS

The only separators allowed in the comment are the space, tab, and comma.

Example:

```
1:  Rem  This file checks new disks
2:  Rem  It is named NEWDISK.BAT
3:  Pause  Insert new disk in drive B
4:  Format B:/s
6:  Chkdsk B:
```

You can use Rem without a comment in your file to add spacing for readability.

I

NAME

Shift

PURPOSE

Allows access to more than 10 replaceable parameters in batch file processing.

SYNTAX

```
shift
```

COMMENTS

Usually, command files are limited to handling 10 parameters, %0 through %9. To allow access to more than ten parameters, use Shift to change the command line parameters. For example:

```
if:     %0 = "tan"
        %1 = "bar"
        %2 = "name"
        %3...%9 are empty
```

then a Shift results in the following:

```
        %0 = "bar"
        %1 = "name"
        %2...%9 are empty
```

If there are more than 10 parameters given on a command line, those that appear after the 10th (%9) will be shifted one at a time into %9 by successive shifts.

Note

There is no backward shift. Once Shift is executed, the 0 parameter (%0) that existed before the shift cannot be recovered.

Chapter 4
MS-DOS Editing and Function Keys

CHAPTER 4

MS-DOS EDITING AND FUNCTION KEYS

4.1 SPECIAL MS-DOS EDITING KEYS

The special editing keys deserve particular emphasis because they depart from the way in which most operating systems handle command input. You do not have to type the same sequences of keys repeatedly, because the last command line is automatically placed in a special storage area called a template.

By using the template and the special editing keys, you can take advantage of the following MS-DOS features:

1. A command line can be instantly repeated by pressing two keys.

2. If you make a mistake in the command line, you can edit it and retry without having to retype the entire command line.

3. A command line that is similar to a preceding command line can be edited and executed with a minimum of typing by pressing a special editing key.

The relationship between the command line and the template is shown in Figure 4.1.

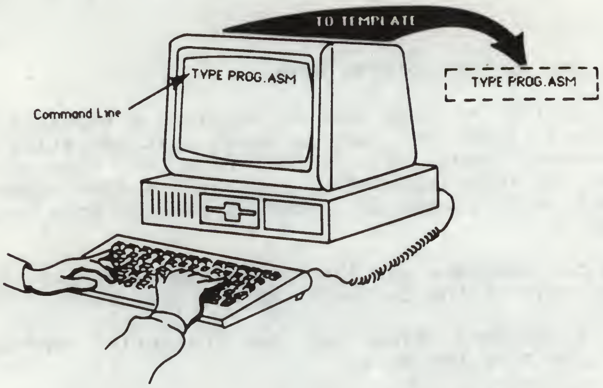


Figure 4.1. Command Line and Template

As seen in Figure 4.1, you type a command to MS-DOS on the command line. When you press the Return key, the command is automatically sent to the command processor (COMMAND.COM) for execution. At the same time, a copy of this command is sent to the template. You can now recall the command or modify it with MS-DOS special editing keys.

Table 4.1 contains a complete list of the special editing keys. Each of these keys is more fully described in Chapter 5, "The Line Editor (EDLIN)."

Note

The keys on your keyboard may not correspond to the specific keys in the following examples. Therefore, these MS-DOS editing keys will be referred to by FUNCTION rather than by name. When an example says to press the <SKIP1> key, find the key on your keyboard that corresponds to the "skip one character" editing function and press it.

Some functions may require you to press two keys. Consult the operating manual for your terminal to determine which keys correspond to the MS-DOS editing functions described below.

You may wish to write in the keys on your keyboard that correspond to the editing keys described in the following table. Space is provided to do this.

Table 4.1 Special Editing Functions

Key	**Editing Function**	**Your Keyboard**
<COPY1>	Copies one character from the template to the command line	
<COPYUP>	Copies characters up to the character specified in the template and puts these characters on the command line	
<COPYALL>	Copies all remaining characters in the template to the command line	
<SKIP1>	Skips over (does not copy) a character in the template	
<SKIPUP>	Skips over (does not copy) the characters in the template up to the character specified	
<VOID>	Voids the current input; leaves the template unchanged	
<INSERT>	Enters/exits insert mode	
<NEWLINE>	Makes the new line the new template	
<Control-Z>	Puts a Control-Z (1AH) end-of-file character in the new template	

Examples:

If you type:

```
     dir prog.com
```

MS-DOS displays information about the file PROG.COM on your screen. The command line (dir prog.com) is also saved in the template. To repeat the command, just press two keys: <COPYALL> and Return.

The repeated command is displayed on the screen as you type, as shown below:

```
     <COPYALL>dir prog.com<RETURN>
```

Notice that pressing the <COPYALL> key causes the contents of the template to be copied to the command line; pressing the Return key causes the command line to be sent to the command processor for execution.

If you want to display information about a file named PROG.ASM, you can use the contents of the template and type:

```
     <COPYUP>c
```

Typing <COPYUP>C copies all characters from the template to the command line, up to but not including C. MS-DOS displays:

```
     DIR PROG._
```

Note that the underline is your cursor. Now type:

```
     asm
```

The result is:

```
     DIR PROG.ASM_
```

The command line DIR PROG.ASM is now in the template and ready to be sent to the command processor for execution. To run the command, press the Return key.

Now, assume that you want to run the following command:

```
     TYPE PROG.ASM
```

To do this, type:

```
     type <INSERT> <COPYALL><RETURN>
```

Notice that when you are typing, the characters are entered directly into the command line and overwrite corresponding characters in the template. This automatic replacement feature is turned off when you press the <INSERT> key. Thus, the characters "TYPE" replace the characters "DIR " in the template. To insert a space between "TYPE" and "PROG.ASM", you pressed <INSERT> and then the Spacebar. Finally, to copy the rest of the template to the command line, you pressed <COPYALL> and then the Return key. The command TYPE PROG.ASM has been processed by MS-DOS, and the template becomes TYPE PROG.ASM.

If you had misspelled TYPE as BYTE, a command error would have occurred. Still, instead of throwing away the whole command, you could save the misspelled line before you press the Return key by creating a new template with the <NEWLINE> key:

```
BYTE PROG.ASM<NEWLINE>
```

You could then edit this error by typing:

```
T<COPY1>P<COPYALL>
```

The <COPY1> key copies a single character from the template to the command line. The resultant command line is then the command that you want:

```
TYPE PROG.ASM
```

As an alternative, you can use the same template containing BYTE PROG.ASM and then use the <SKIP1> and <INSERT> keys to get the same result:

```
<SKIP1><SKIP1><COPY1><INSERT>YP<COPYALL>
```

To illustrate how the command line is affected as you type, examine the keys pressed on the left; their effect on the command line is shown on the right:

<SKIP1>	_	Skips over 1st template character
<SKIP1>		Skips over 2nd template character
<COPY1>	T	Copies 3rd template character
<INSERT>YP	TYP	Inserts two characters
<COPYALL>	TYPE PROG.ASM	Copies rest of template

Notice that <SKIP1> does not affect the command line. It affects the template by deleting the first character. Similarly, <SKIPUP> deletes characters in the template, up to, but not including, a given character.

These special editing keys can add to your effectiveness at the keyboard. The next section describes control character functions that can also help when you are typing commands.

4.2 CONTROL CHARACTER FUNCTIONS

A control character function is a function that affects the command line. You have already learned about Control-C and Control-S. Other control character functions are described below. (Note that some control character functions may not apply to your computer.)

Remember that when you type a control character sequence, such as Control-C, you must hold down the control key and then press the C key.

Table 4.2 Control Character Functions

Control Character	Function
Control-C	Aborts current command.
Control-H	Removes last character from command line, and erases character from terminal screen.
Control-J	Inserts physical end-of-line, but does not empty command line. Use the Linefeed key to extend the current logical line beyond the physical limits of the terminal screen.
Control-N	Echoes output to a lineprinter.
Control-P	Sends terminal output to a lineprinter.
Control-S	Suspends output display on the screen. Press Control-S to resume.
Control-X	Cancels the current line; empties the command line; and then outputs a backslash (\\), return, and linefeed. The template used by the special editing commands is not affected.

Chapter 5
.EXE File Structure and Loading

CHAPTER 5

THE LINE EDITOR (EDLIN)

5.1 INTRODUCTION

In this chapter, you will learn how to use EDLIN, the line editor program. You can use EDLIN to create, change, and display files, whether they are program or text files.

You can use EDLIN to:

- Create new files and save them on disk.
- Update existing files and save both the updated and original files.
- Delete, edit, insert, and display lines in files.
- Search for, delete, or replace text within one or more lines in a file.

EDLIN divides all text into lines, and each line can have up to 253 characters. EDLIN gives each line a line number. Although you will see these line numbers on the screen when you use EDLIN, they are not actually saved with the file.

When you insert lines, the line numbers after the inserted text are adjusted automatically. When you delete lines in a file, all line numbers following the deleted text are renumbered automatically. Lines are always numbered consecutively in your file.

5.2 HOW TO START EDLIN

To start EDLIN, type:

```
EDLIN <filename>
```

If you are creating a new file, the filename should be the name or path of the file you wish to create. If EDLIN does not find this file on a disk drive, EDLIN will create a new file with the name or path you specify. If you are creating a new file, you will see this when EDLIN starts:

```
New file
*_
```

Notice that the prompt for EDLIN is an asterisk (*).

You can now type lines of text into your new file. To begin entering text, you must type an I (Insert) command to insert lines. The I command is discussed later in this chapter.

If you want to edit an existing file, <filespec> should be the name of the file you want to edit. When EDLIN finds the file you specify, the file will be loaded into memory. If the entire file can be loaded, EDLIN will display the following message:

```
End of input file
*
```

You can then edit the file using EDLIN editing commands.

If the file is too large to be loaded into memory, EDLIN will load lines until memory is 3/4 full, then display the * prompt. You can then edit the portion of the file that is in memory.

To edit the rest of the file, you must save some of the edited lines on disk to free memory; then EDLIN can load the unedited lines from disk into memory. Refer to the Write and Append commands in this chapter for instructions on editing large files.

Do not try to edit a file with a filename extension of .BAK because EDLIN assumes that any .BAK file is a backup file. If you need to edit such a file, rename the file with another extension (using the MS-DOS Ren command discussed in Chapter 3), then start EDLIN and specify the new filename.

5.3 HOW TO QUIT EDLIN

When you finish the editing session, you can save the original and the updated (new) files by using the End command. The End command is discussed in this chapter in the section 5.7, "EDLIN Commands." The original file is renamed with an extension of .BAK, and the new file has the filename and extension you specify when you start EDLIN. The original .BAK file will not be erased until the end of the editing session, or until EDLIN needs the disk space.

5.4 SPECIAL EDITING KEYS

The special editing keys and template discussed in Chapter 4 can be used to edit your text files. These keys are discussed in detail in this section.

Table 5.1 summarizes the commands, codes, and functions. Descriptions of the special editing keys follow the table.

Note

The keys on your keyboard may not correspond to the specific keys in the following examples. Therefore, these MS-DOS editing keys will be referred to by FUNCTION rather than by name. When an example says to press the <SKIP1> key, find the key on your keyboard that corresponds to the "skip one character" editing function and press it. Some functions may require you to press two keys. Consult the operating manual for your computer to determine which keys correspond to the MS-DOS editing functions described here.

Table 5.1 Special Editing Keys

Function	Key	Description
Copy one character	<COPY1>	Copies one character from the template to the new line.
Copy up to character	<COPYUP>	Copies all characters from the template to the new line, up to the character specified.
Copy template	<COPYALL>	Copies all remaining characters in the template to the screen.
Skip one character	<SKIP1>	Does not copy (skips over) a character.
Skip up to character	<SKIPUP>	Does not copy (skips over) the characters in the template, up to the character specified.
Quit input	<VOID>	Voids the current input; leaves the template unchanged.
Insert mode	<INSERT>	Enters/exits insert mode.
Replace mode	<REPLACE>	Turns insert mode off; this is the default.
New template	<NEWLINE>	Makes the new line the new template.

KEY

<COPY1>

PURPOSE

Copies one character from the template to the current line.

COMMENTS

Pressing the <COPY1> key copies one character from the template to the current line. When you press <COPY1>, EDLIN inserts one character in the current line and turns off insert mode.

Example:

Assume that the screen shows:

```
1:*This is a sample file.
2:*_
```

At the beginning of the editing session, the cursor (shown by the underline) is at the beginning of the line. Pressing the <COPY1> key copies the first character (T) to the second of the two lines displayed:

```
        1:*This is a sample file.
<COPY1> 2:*T_
```

Each time you press the <COPY1> key, one more character appears:

```
<COPY1> 1:*Th_
<COPY1> 2:*Thi_
<COPY1> 2:*This_
```

KEY

<COPYUP>

PURPOSE

Copies multiple characters up to a given character.

COMMENTS

Pressing the <COPYUP> key copies all characters up to a given character from the template to the current line. The given character is the next character typed after <COPYUP>; it is not copied or displayed on the screen. Pressing the <COPYUP> key moves the cursor to the single character that is specified in the command. If the template does not contain the character, nothing is copied. Pressing <COPYUP> also automatically turns off insert mode.

Example:

Assume that the screen shows:

```
1:*This is a sample file.
2:*_
```

At the beginning of the editing session, the cursor (shown by the underline) is at the beginning of the line. Pressing the <COPYUP> key copies all characters up to the character specified right after the <COPYUP> key.

```
            1:*This is a sample file
<COPYUP>p 1:*This is a sam_
```

KEY

<COPYALL>

PURPOSE

Copies the template to the current line.

COMMENTS

Pressing the <COPYALL> key copies all remaining characters from the template to the current line. No matter where the cursor is when the <COPYALL> key is pressed, EDLIN displays the rest of the line, and the cursor appears at the end of the line.

Example:

Assume that the screen shows:

```
     1:*This is a sample file.
     2:*_
```

At the beginning of the editing session, the cursor (shown by the underline) is at the beginning of the line. Pressing the <COPYALL> key copies all characters from the template (shown in the upper line displayed) to the line with the cursor (the lower line displayed):

```
          1:*This is a sample file   (template)
<COPYALL> 2:*This is a sample file._ (current
                                       line)
```

Also, insert mode is automatically turned off.

KEY

<SKIP1>

PURPOSE

Skips over one character in the template.

COMMENTS

Pressing the <SKIP1> key skips over one character in the template. Each time you press the <SKIP1> key, one character is not copied from the template. The action of the <SKIP1> key is similar to the <COPY1> key, except that <SKIP1> skips a character in the template rather than copying it to the current line.

Example:

Assume that the screen shows:

```
     1:*This is a sample file.
     2:*_
```

At the beginning of the editing session, the cursor (shown by the underline) is at the beginning of the line. Pressing the <SKIP1> key skips over the first character (T).

```
              1:*This is a sample file
      <SKIP1> 2:*_
```

The cursor does not move and only the template is changed. To see how much of the line has been skipped over, press the <COPYALL> key, which moves the cursor past the last character of the line.

```
              1:*This is a sample file.
      <SKIP1> 2:*
    <COPYALL> 2:*his is a sample file._
```

KEY

<SKIPUP>

PURPOSE

Skips multiple characters in the template up to the specified character.

COMMENTS

Pressing the <SKIPUP> key skips over all characters up to a given character in the template. This character is not copied and does not appear on the screen. If the template does not contain the character, nothing is skipped over. The action of the <SKIPUP> key is similar to the <COPYUP> key, except that <SKIPUP> skips over characters in the template rather than copying them to the current line.

Example:

Assume that the screen shows:

```
1:*This is a sample file.
2:*_
```

At the beginning of the editing session, the cursor (shown by the underline) is at the beginning of the line. Pressing the <SKIPUP> key skips over all the characters in the template up to the character pressed after the <SKIPUP> key (in this example, "p"):

```
            1:*This is a sample file.
<SKIPUP>p 2:*_
```

The cursor does not move. To see how much of the line has been skipped over, press the <COPYALL> key to copy the template. This moves the cursor to the end of the line:

```
                     1:*This is a sample file.
<SKIPUP>p<COPYALL>   2:*ple file._
```

KEY

<VOID>

PURPOSE

Quits input and empties the current line.

COMMENTS

Pressing the <VOID> key empties the current line, but it leaves the template unchanged. <VOID> also prints a back slash (\), carriage return, and line feed, and turns insert mode off. The cursor (shown by the underline) is at the beginning of the line. Pressing the <COPYALL> key copies the template to the current line and the current line appears as it was before <VOID> was pressed.

Example:

Assume that the screen shows:

```
1:*This is a sample file.
2:*_
```

At the beginning of the editing session, the cursor is at the beginning of the line. Assume that line 2 contains the words "Sample File":

```
1:*This is a sample file.
2:*Sample File_
```

To cancel line 2 (Sample File), and to keep "This is a sample file.", press <VOID>. Notice that a backslash appears on the Sample File line to tell you it has been canceled.

```
       1:*This is a sample file.
<VOID> 2:*Sample File\
```

Press the Return key to keep line 1, or to perform any other editing functions. If you press <COPYALL>, EDLIN copies the original template to the line.

```
          1: This is a sample file.
<VOID>    2: Sample File.\
<COPYALL> 2: This is a sample file.
```

KEY

<INSERT>

PURPOSE

Enters/exits insert mode.

COMMENTS

Pressing the <INSERT> key causes EDLIN to enter or exit insert mode. The cursor in the template does not move. The cursor in the current line moves as each character is inserted. However, when you finish inserting characters, the cursor is at the same character in the template as it was before the insertion began. Characters are inserted into the template In front of the cursor.

Example:

Assume that the screen shows:

```
1:*This is a sample file.
2:*_
```

At the beginning of the editing session, the cursor (shown by the underline) is at the beginning of the line. Assume that you press the <COPYUP> and f keys:

```
             1:*This is a sample file.
<COPYUP>f 1:*This is a sample _
```

Now press the <INSERT> key and insert the characters "edit" and a space:

```
                1:*This is a sample file.
<COPYUP>f       2:*This is a sample _
<INSERT>edit    2:*This is a sample edit _
```

If you now press the <COPYALL> key, the rest of the template is copied to the line:

```
            1:*This is a sample edit _
<COPYALL> 2:*This is a sample edit file._
```

If, after entering insert mode, you press the Return key, the remainder of the template is cut off, and the current line ends after the inserted text:

```
2: This is a sample edit
3: *
```

To exit insert mode, simply press the <INSERT> key again.

KEY

<REPLACE>

PURPOSE

Enters replace mode.

COMMENTS

Pressing the <REPLACE> key causes EDLIN to exit insert mode and to enter replace mode. All the characters you type overstrike and replace characters in the template. When you start to edit a line, replace mode is in effect. If you press the Return key, EDLIN deletes the remainder of the template.

Example:

Assume that the screen shows:

```
     1:*This is a sample file.
     2:*_
```

At the beginning of the editing session, the cursor (shown by the underline) is at the beginning of the line. Assume that you then press <COPYUP>m, <INSERT>lary, <REPLACE>tax, and then <COPYALL>:

```
                      1:*This is a sample file.
     <COPYUP>m        2:*This is a sa_
     <INSERT>lary     2:*This is a salary_
     <REPLACE> tax    2:*This is a salary tax_
     <COPYALL>        2:*This is a salary tax file._
```

Notice that you inserted "lary" and replaced "mple" with " tax". If you type characters that extend beyond the length of the template, the remaining characters in the template are added at the end of the line when you press <COPYALL>.

KEY

<NEWLINE>

PURPOSE

Creates a new template.

COMMENTS

Pressing the <NEWLINE> key copies the current line to the template. The contents of the previous template are deleted. Pressing <NEWLINE> displays an @ ("at sign" character), and outputs a return and a line feed. When you press <NEWLINE>, EDLIN empties the current line and turns off insert mode.

Note

<NEWLINE> performs the same function as the <VOID> key, except that the template is changed and an @ ("at sign" character) is printed instead of a \ (backslash).

Example:

Assume that the screen shows:

```
1:*This is a sample file.
2:*_
```

At the beginning of the editing session, the cursor (shown by the underline) is at the beginning of the line. Assume that you enter <COPYUP>m, <INSERT>lary, <REPLACE>tax, and then <COPYALL>:

```
                  1:*This is a sample file.
<COPYUP>m         2:*This is a sa_
<INSERT>lary      2:*This is a salary_
<REPLACE> tax     2:*This is a salary tax_
<COPYALL>         2:*This is a salary tax file._
```

At this point, assume that you want this line to be the new template, so you press the <NEWLINE> key:

```
1: This is a sample file.
2: This is a salary tax file.@
```

The @ indicates that this new line is now the new template. Additional editing can be done using the new template.

5.5 COMMAND INFORMATION

EDLIN commands are shown in the following table. They are also described in this chapter following the description of command options.

Table 5.2. EDLIN Commands

Command	What it does...
<line>	Edits a line or lines
A	Appends lines
C	Copies lines
D	Deletes lines
E	Ends editing
I	Inserts lines
L	Lists text
M	Moves lines
P	Pages text
Q	Quits editing
R	Replaces lines
S	Searches text
T	Transfers text
W	Writes lines

EDLIN commands perform editing functions on lines of text. The following list contains information you should read before you use EDLIN commands.

1. Pathnames are acceptable in commands. For example, by typing:

   ```
   edlin \user\joe\report.may
   ```

   you can edit the REPORT.MAY file in the subdirectory JOE.

2. You can reference line numbers relative to the current line (the line with the asterisk). Use a minus sign with a number to indicate lines before the current line. Use a plus sign with a number to indicate lines after the current line.

Note

A capital "L" has been used here for the "list" command in order to avoid confusion with the number one. A small letter "l" would work just as well.

Example:

```
-10,+10L
```

This command lists 10 lines before the current line, the current line, and 10 lines after the current line.

3. Multiple commands may be typed on one command line. When you type a command to edit a single line using a line number (<line>), a semicolon must separate commands on the line. Otherwise, one command may follow another without any special separators. In the case of a Search or Replace command, the string may be ended by pressing Control-Z instead of the Return key.

   Examples:

   The following command line edits line 15 and then displays lines 10 through 20 on the screen.

   ```
   15;-5,+5L
   ```

   The command line in the next example searches for the words "This house," and then displays five lines before and five lines after the line that contains "This house." If EDLIN does not find any lines that contain the words "This house", then the displayed lines are those line numbers relative to the current line. Note that the current line (the line with the asterisk) must be <u>before</u> the line or lines that contain the search string.

   ```
   sThis house <Control-Z>-5,+5L
   ```

4. You can type EDLIN commands <u>with</u> or <u>without</u> a space between the line number and command. For example, to delete line 6, the command 6d is the same as 6 d.

5. You can insert a control character (such as Control-C) into text by using the quote character Control-V before it while in insert mode. Control-V tells MS-DOS to recognize the next <u>capital</u> letter typed as a control character. It is also possible to use a control character in any Search or Replace command by using the special quote character. For example:

   ```
   s<Control-V>Z
   will find the first occurrence
   of Control-Z in a file

   r<Control-V>Z<Control-Z>dog
   will replace all occurrences
   of Control-Z in a file by dog

   s<Control-V>C<Control-Z>cat
   will replace all occurrences
   of Control-C by cat
   ```

   It is possible to insert Control-V into the text by typing Control-V-V.

6. The Control-Z character ordinarily tells EDLIN, "This is the end of the file." If you have Control-Z characters elsewhere in your file, you must tell EDLIN that these other control characters do not mean end-of-file. Use the /b switch to tell EDLIN to ignore any Control-Z characters in the file and to show you the entire file. For example, when you start EDLIN, type:

   ```
   EDLIN <filename> /b
   ```

   to ignore all Control-Z characters.

5.6 COMMAND OPTIONS

Many EDLIN commands accept one or more options. The effect of a command option varies, depending on with which command it is used. The following list describes each option.

<line> <line> indicates a line number that you type. Use a comma or a space to separate the numbers from other line numbers, other options, and from the command.

<line> may be specified one of three ways:

<number> Any number less than 65534. If a number larger than the largest existing line number is specified, then <line> means the line after the last line number.

Period (.) If a period is specified for <line>, then <line> means the current line number. The current line is the last line edited, and is not always the last line displayed. EDLIN marks the current line with an asterisk (*) between the line number and the first character.

Pound Sign (#) The pound sign indicates the line after the last line number. If you type # for <line>, this is the same as typing a number larger than the last line number.

Return key If you type a command and then press Return without any of the line markers listed above, EDLIN uses a default value for each command. (Default values may be different for each command).

? The question mark option tells EDLIN to ask you if the correct string has been found. The question mark is used only with the Replace and Search commands. Before continuing, EDLIN waits for you to type a Y or press the Return key for a "yes" response, or to press any other key for a "no" response.

<string>	<string> represents text to be found, to be replaced, or to replace other text. The <string> option is used only with the Search and Replace commands. Each <string> must be ended by a Control-Z or a carriage return (see the Replace command for details). No spaces should be left between strings or between a string and its command letter, unless you want those spaces to be part of the string.

5.7 EDLIN COMMANDS

The following pages describe EDLIN editing commands.

NAME

Append

PURPOSE

Adds the specified number of lines from disk to the file being edited in memory. The lines are added at the end of lines that are currently in memory.

SYNTAX

```
[<n>]a
```

COMMENTS

This command is meaningful only if the file being edited is too large to fit into memory. As many lines as possible are read into memory for editing when you start EDLIN.

To edit the remainder of the file that will not fit into memory, lines that have already been edited must be written to disk. Then you can load unedited lines from disk into memory with the Append command. Refer to the Write command in this chapter for information on how to write edited lines to disk.

Notes

1. If you do not specify the number of lines to append, lines will be appended to memory until available memory is 3/4 full. No action will be taken if available memory is already 3/4 full.
2. EDLIN displays "End of input file" after the Append command reads the last line of the file into memory.

NAME

Copy

PURPOSE

Copies a range of lines to a specified line number. The lines can be copied as many times as you want by using the <count> option.

SYNTAX

```
[<line>],[<line>],<line>[,<count>]c
```

COMMENTS

If you do not specify a number for the <count> option, EDLIN copies the lines one time. If the first or second <line> is omitted, the default is the current line. The file is renumbered automatically after the copy.

The line numbers must not overlap or you will get an "Entry error" message. For example, 3,20,15C would result in an error message.

Examples:

Assume that the following file exists and is ready to edit:

```
1: This is a sample file
2: used to show copying lines.
3: See what happens when you use
4: the Copy command
5: (the C command)
6: to copy text in your file.
```

You can copy this entire block of text by issuing the following command:

```
1,6,7c
```

This will copy lines 1 through 6 and duplicate them one time, beginning on line 7.

The result is:

```
 1: This is a sample file
 2: used to show copying lines.
 3: See what happens when you use
 4: the Copy command
 5: (the C command)
 6: to copy text in your file.
 7: This is a sample file
 8: used to show copying lines.
 9: See what happens when you use
10: the Copy command
11: (the C command)
12: to copy text in your file.
```

If you want to place the text within other text, the third <line> option should specify the line before which you want the copied text to appear. For example, assume that you want to copy lines and insert them within the following file:

```
 1: This is a sample file
 2: used to show copying lines.
 3: See what happens when you use
 4: the Copy command
 5: (the C command)
 6: to copy text in your file.
 7: You can also use Copy
 8: to copy lines of text
 9: to the middle of your file.
10: End of sample file.
```

The command 3,6,10C results in the following file:

```
 1: This is a sample file
 2: used to show copying lines.
 3: See what happens when you use
 4: the Copy command
 5: (the C command)
 6: to copy text in your file.
 7: You can also use Copy
 8: to copy lines of text
 9: to the middle of your file.
10: See what happens when you use
11: the Copy command
12: (the C command)
13: to copy text in your file.
14: End of sample file.
```

NAME

Delete

PURPOSE

Deletes a specified range of lines in a file.

SYNTAX

```
[<line>][,<line>]d
```

COMMENTS

If the first <line> is omitted, that option will default to the current line (the line with the asterisk next to the line number). If the second <line> is omitted, then just the first <line> will be deleted. When lines have been deleted, the line immediately after the deleted section becomes the current line and has the same line number as the first deleted <line> had before the deletion occurred.

Examples:

Assume that the following file exists and is ready to edit:

```
 1: This is a sample file
 2: used to show dynamic line numbers.
 3: See what happens when you use
 4: Delete and Insert
   .
   .
   .
25: (the D and I commands)
26: to edit the text
27:*in your file.
```

To delete multiple lines, type <line>,<line>d:

```
5,24d
```

The result is:

```
1: This is a sample file
2: used to show dynamic line numbers.
3: See what happens when you use
4: Delete and Insert
5: (the D and I commands)
6: to edit text
7:*in your file.
```

To delete a single line, type:

```
6d
```

The result is:

```
1: This is a sample file
2: used to show dynamic line numbers.
3: See what happens when you use
4: Delete and Insert
5: (the D and I commands)
6:*in your file.
```

Next, delete a range of lines from the following file:

```
1: This is a sample file
2: used to show dynamic line numbers.
3:*See what happens when you use
4: Delete and Insert
5: (the D and I commands)
6: to edit text
7: in your file.
```

To delete a range of lines beginning with the current line (in this case, line number 3), type:

```
,6d
```

The result is:

```
1: This is a sample file
2: used to show dynamic line numbers.
3:*in your file.
```

Notice that the lines are automatically renumbered.

NAME

Edit

PURPOSE

Edits a line of text.

SYNTAX

[<line>]

COMMENTS

When a line number is typed, EDLIN displays the line number and text; then, on the line below, EDLIN reprints the line number. The line is now ready for editing. You may use any of the EDLIN editing commands to edit the line. The existing text of the line serves as the template until you press the Return key. If you do not type a line number (that is, if only the Return key is pressed), the line after the current line (marked with an asterisk (*)) is edited. If no changes to the current line are needed and the cursor is at the beginning or end of the line, press the Return key to accept the line as is.

Warning

If you press the Return key while the cursor is in the middle of the line, EDLIN deletes the remainder of the line.

Example:

Assume that the following file exists and is ready to edit:

```
1: This is a sample file
2: used to show
3: the editing of line
4:*four.
```

To edit line 4, type:

```
4
```

The contents of the line are displayed with a cursor below the line:

```
4:* four.
4:*_
```

Now, using the <COPYALL> special editing key, type:

```
<INSERT>number      4: number_
<COPYALL><RETURN> 4: number¯four.
```

NAME

End

PURPOSE

Ends the editing session.

SYNTAX

```
e
```

COMMENTS

This command saves the edited file on disk, renames the original input file <filename>.BAK, and then exits EDLIN. If the file was created during the editing session, no .BAK file is created.

The E command takes no options. Therefore, you cannot tell EDLIN on which drive to save the file. The drive you want to save the file on must be selected when the editing session is started. Type the drive letter in front of the filename you specify when you start EDLIN. If the drive is not selected when EDLIN is started, the file will be saved on the disk in the default drive. You can still copy the file to a different drive using the MS-DOS Copy command.

Make sure that the disk contains enough free space for the entire file. If the disk does not contain enough free space, the write will be aborted and the edited file lost, although part of the file might be written out to the disk.

Example:

```
e<RETURN>
```

After EDLIN processes the E command, the screen displays the MS-DOS prompt (for example, A>).

NAME

Insert

PURPOSE

Inserts text immediately before the specified <line>.

SYNTAX

```
[<line>]i
```

COMMENTS

If you are creating a new file, the I command must be typed before text can be inserted. Text begins with line number 1. Successive line numbers appear automatically each time you press the Return key.

EDLIN remains in insert mode until Control-C is typed. When the insert is completed and insert mode has been exited, the line immediately following the inserted lines becomes the current line. All line numbers following the inserted section are incremented by the number of lines inserted.

If <line> is not specified, the default is the current line number and the lines are inserted before the current line. If <line> is a number larger than the last line number, or if a pound sign (#) is specified as <line>, the inserted lines are appended to the end of the file. In this case, the last line inserted becomes the current line.

Examples:

Assume that the following file exists and is ready to edit:

```
1: This is a sample file
2: used to show dynamic line numbers.
3: See what happens when you use
4: Delete and Insert
5: (the D and I commands)
6: to edit text
7:*in your file.
```

To insert text before a specific line , type:

```
7i
```

The result is:

```
7:*_
```

Now, type the new text for line 7:

```
7: and renumber lines
```

Then, to end the insertion, press Control-C on the next line:

```
8: <Control-C>
```

Now type L to list the file. The result is:

```
1: This is a sample file
2: used to show dynamic line numbers.
3: See what happens when you use
4: Delete and Insert
5: (the D and I commands)
6: to edit text
7. and renumber lines
8:*in your file.
```

To insert lines immediately before the current line, type:

```
i
```

The result is:

```
8:*_
```

Now, insert the following text and terminate with a Control-C on the next line:

```
8: so they are consecutive
9: <Control-C>
```

Now to list the file and see the result, type:

```
L
```

The result is:

```
 1: This is a sample file
 2: used to show dynamic line numbers.
 3: See what happens when you use
 4: Delete and Insert
 5: (the D and I commands)
 6: to edit text
 7: and renumber lines
 8: so they are consecutive
 9:*in your file.
```

To append new lines to the end of the file, type:

```
10i
```

This produces the following:

```
10:*_
```

Now, type the following new lines:

```
10: The Insert command can place new lines
11: in the file; there's no problem
12: because the line numbers are dynamic;
13: they'll go all the way to 65533.
```

End the insertion by pressing Control-C on line 14. The new lines will appear at the end of all previous lines in the file. Now list all the lines:

```
1, 13L
```

The result is:

```
 1: This is a sample file
 2: used to show dynamic line numbers.
 3: See what happens when you use
 4: Delete and Insert
 5: (the D and I commands)
 6: to edit text
 7: and renumber lines
 8: so they are consecutive
 9: in your file.
10: The insert command can place new lines
11: in the file; there's no problem
12: because the line numbers are dynamic;
13: they'll go all the way to 65533.
```

NAME

List

PURPOSE

Lists a range of lines, including the two lines specified.

SYNTAX

```
[<line>][,<line>]L
```

COMMENTS

A capital letter "L" has been used here to avoid confusion with the number one. A small letter "l" would work just as well.

Default values are provided if you do not type either option. If you do not type the first option, as in:

```
,<line>L
```

the display will start 11 lines before the current line and end with the specified <line>. The beginning comma is required to indicate the omitted first option.

Note

If the specified <line> is more than 11 lines before the current line, the display will be the same as if you omitted both options.

If you omit the second option, as in:

```
<line>L
```

23 lines will be displayed, starting with the specified <line>. If you just type:

```
L
```

23 lines will be displayed--the 11 lines before the current line, the current line, and the 11 lines after the current line. If there are less than 11 lines before the current line, more than 11 lines after the current line will be displayed to make a total of 23 lines.

Examples:

Assume that the following file exists and is ready to edit:

```
     1: This is a sample file
     2: used to show dynamic line numbers.
     3: See what happens when you use
     4: Delete and Insert
     5: (the D and I commands)
      .
      .
      .
    15:*The current line contains an asterisk.
      .
      .
      .
    26: to edit text
    27: in your file.
```

To list a range of lines without reference to the current line, type <line>,<line>L:

```
2,5L
```

The result is:

```
2: used to show dynamic line numbers.
3: See what happens when you use
4: Delete and Insert
5: (the D and I commands)
```

To list a range of lines beginning with the current line, type <line>, L (where <line> is the current line):

```
15, L
```

The result is:

```
15:*The current line contains an asterisk.
  .
  .
  .
26: to edit text
27: in your file.
```

To list a range of 23 lines centered around the current line, type only L:

```
     L
```

The result is:

```
4: Delete and Insert
5: (the D and I commands)
 .
 .
 .
13: The current line is listed in the middle.
14: The current line remains unchanged.
15:*The current line contains an asterisk.
 .
 .
 .
26: to edit text
27: in your file.
```

NAME

Move

PURPOSE

Moves a range of text to the line specified.

SYNTAX

```
[<line>],<line>,<line>m
```

COMMENTS

Use the Move command to move a block of text to another location in the file. The lines are renumbered according to the direction of the move. For example:

```
,+25,100m
```

moves the text from the current line plus 25 lines to line 100. If the line numbers overlap, EDLIN displays an "Entry error" message.

To move lines 20-30 to line 100, type:

```
20,30,100m
```

NAME

Page

PURPOSE

Pages through a file 23 lines at a time.

SYNTAX

```
[<line>][,<line>]p
```

COMMENTS

If you do not type the first <line>, that number will default to the current line plus one. If you do not type the second <line> option, 23 lines will be listed. The new current line becomes the last line displayed and is marked with an asterisk.

NAME

Quit

PURPOSE

Quits the editing session, does <u>not</u> save any editing changes, and exits to the MS-DOS operating system.

SYNTAX

```
q
```

COMMENTS

EDLIN prompts you to make sure you don't want to save the changes.

Type Y if you want to quit the editing session. EDLIN does not save any editing changes and does not create a backup (.BAK) file. Refer to the End command in this chapter for information about the .BAK file.

Type N if you want to continue the editing session.

Note

When started, EDLIN erases any previous copy of the file with an extension of .BAK to make room to save the new copy. If you reply Y to the "Abort edit (Y/N)?" message, EDLIN deletes your previous backup copy.

```
Example:  q
          Abort edit (Y/N)?y<Return>
       A>_
```

NAME

Replace

PURPOSE

Replaces all occurrences of a string of text in a range with a different string of text (or blanks).

SYNTAX

```
[<line>][,<line>][?]r<string1><Control-Z>
    <string2>
```

COMMENTS

Each time EDLIN finds <string1>, it replaces the text with <string2>. EDLIN displays each line that changes. If a line contains two or more replacements, then the line is displayed once for each change. When all changes are made, the Replace command ends and the asterisk prompt reappears.

If you want to replace one string of text with another string, the two strings must be separated by a Control-Z. The second string can be ended by pressing the Return key.

If you do not specify string1, the Replace command assumes the "old" (any previous) value. If this is the first Replace done during the session, and you do not specify string1, the command ends. If you do not specify string2, end string1 with a carriage return.

If you omit the first line option, EDLIN will select a default line that is one line <u>after</u> the current line. The default for the second line argument is #. Remember that # means the line after the last line of the file.

If you end string1 with a Control-Z and do not specify string2, EDLIN assumes you want blank spaces for string2. For example:

```
r dogs<Control-Z><RETURN>
```

deletes all occurrences of "dogs", but

```
r dogs<RETURN>
r<RETURN>
```

replaces "dogs" with the previous string2, and the previous string1 becomes the previous string2. Note that "previous" here refers to an earlier string specified either in a Search or a Replace command.

If you specify a question mark (?) in the Replace command, the Replace stops at each line with a string that matches string1, displays the line with string2, and then displays the prompt "O.K.?". If you press Y or Return, string2 replaces string1, and the next occurrence of string1 is displayed. EDLIN displays the "O.K.?" prompt again. This process continues until the end of the range or until the end of the file. After the last string1 is found, EDLIN displays the asterisk prompt.

If you press any key besides Y or the Return key after the "O.K.?" prompt, the string1 will be left as it was, and Replace will go to the next occurrence of string1. If string1 occurs more than once in a line, each occurrence of string1 is replaced individually, and the "O.K.?" prompt is displayed after each replacement. In this way, only the desired string1 is replaced, and you can prevent unwanted substitutions.

Examples:

Assume that the following file exists and is ready for editing:

```
 1: This is a sample file
 2: used to show dynamic line numbers.
 3: See what happens when you use
 4: Delete and Insert
 5: (the D and I commands)
 6: to edit text
 7: in your file.
 8: The insert command can place new lines
 9: in the file; there's no problem
10: because the line numbers are dynamic;
11: they'll go all the way to 65533.
```

To replace all occurrences of "and" with "or" in lines 2 through 10, type:

```
     2,12 rand<Control-Z>or
```

and press the Return key. The result is:

```
     4: Delete or Insert
     5: (the D or I commands)
     5: (the D or I commors)
     8: The insert commor can place new lines
```

(Notice that line 5 has two replacements and is shown twice).

In the above example, some unwanted changes occurred. To avoid these and to confirm each replacement, the same file can be used with a slightly different command.

In the next example, to replace only certain occurrences of "and" with "or", type:

```
     1, 15? rand<Control-Z>or
```

and press the Return key. The result is:

```
     4: Delete or Insert
     O.K.? Y
     5: (The D or I commands)
     O.K.? Y
     5: (The D or I commors)
     O.K.? N
     8: The insert commor can place new lines
     O.K.? N
     *_
```

Now, type the List command (L) to see the result of all these changes:

```
        .
        .
     4: Delete or Insert
     5: (The D or I commands)
        .
     8: The Insert command can place new lines
        .
        .
```

NAME

Search

PURPOSE

Searches a range of lines for a string of text.

SYNTAX

```
[<line>][,<line>][?]s<string><RETURN>
```

COMMENTS

The string must end with a return. EDLIN displays the first line that matches the string and that line becomes the current line. Unless you type the question mark (?) option, the Search command ends when it finds the first match. If EDLIN cannot find a line with a match, the message "Not found" is displayed.

If you include the question mark option (?), EDLIN displays the first line with a matching string; it will then prompt you with the message O.K.?. If you press either the Y or the Return key, the line becomes the current line and the search ends. If you press any other key, the search continues until another match is found, or until all lines have been searched. (The search ends when EDLIN displays the "Not found" message.)

If you do not type the first line number, EDLIN selects the line after the current line. If you do not type the second line number, the second line defaults to # (the line after the last line of the file). If you omit a search string, EDLIN uses the text from any previous Search or Replace command. If this is the first Search or Replace command you've used this session, and you do not specify a search string, the Search command ends immediately.

Examples:

Assume that the following file exists and is ready for editing:

```
 1: This is a sample file
 2: used to show dynamic line numbers.
 3: See what happens when you use
 4: Delete and Insert
 5: (the D and I commands)
 6: to edit text
 7: in your file.
 8: The insert command can place new lines
 9: in the file; there's no problem
10: because the line numbers are dynamic;
11:*they'll go all the way to 65533.
```

To search for the first occurrence of "and", type

```
2,12 sand
```

and press the Return key. EDLIN displays the following lines:

```
4: Delete and Insert
```

To get the "and" in line 5, modify the search command by typing:

```
<SKIP1><COPYALL>,12 sand
```

and press the Return key. The search continues from the line after the current line (line 4), since no first line was given. The result is:

```
5: (the D and I commands)
```

To search through several occurrences of a string until the correct string is found, type:

```
1, ? sand
```

The result is:

```
4: Delete and Insert
O.K.?_
```

If you press any key (except Y or the Return key), the search continues, so type N here:

```
O.K.? n
```

Continue:

```
5: (the D and I commands)
O.K.?_
```

Now press Y to terminate the search:

```
O.K.? Y
*_
```

To search for string XYZ without the verification (O.K.?), type:

```
Sxyz
```

and press the Return key.

EDLIN reports a match and continues to search for the same string when you retype the Search command and press Return:

```
s<RETURN>
```

EDLIN reports another match, if there is one.

```
s<RETURN>
```

EDLIN displays "Not found" at the end of the search.

Note that the text string defaults to any string specified by a previous Replace or Search command.

NAME

Transfer

PURPOSE

Inserts (merges) the contents of a file into the file currently being edited at a specific line number. If the line number is omitted, then the current line is used.

SYNTAX

```
[<line>]t<pathname>
```

COMMENTS

This command is useful if you want to put the contents of one file into another file, or into the text you are typing. The transferred text is inserted at the line number specified by the <line> option and the lines are renumbered.

NAME

Write

PURPOSE

Writes a specific number of lines to disk from the lines that are being edited in memory. Lines are written to disk beginning with line number 1.

SYNTAX

```
[<n>]w
```

COMMENTS

This command is meaningful only if the file you are editing is too large to fit into memory. When you start EDLIN, EDLIN reads lines into memory until memory is 3/4 full.

To edit the remainder of your file, you must write edited lines in memory to disk. Then you can load additional unedited lines from disk into memory by using the Append command.

Note

If you do not specify the number of lines, lines will be written until memory is 3/4 full. EDLIN will not write any lines to disk if memory is already more than 3/4 full. All lines are renumbered, so that the first remaining line becomes line number 1.

5.8 ERROR MESSAGES

EDLIN displays the following error messages:

Cannot edit .BAK file--rename file

Cause: You attempted to edit a file with a filename extension of .BAK. .BAK files cannot be edited because EDLIN reserves this extension for backup copies.

Cure: If you need to edit a file with an extension of .BAK, you must either rename the file with a different extension, or copy the .BAK file and give it a different filename extension. Refer to Chapter 3 for information on the Rename and Copy commands, if you wish to do this.

Disk Full--file write not completed

Cause: You gave the End command, but the disk did not contain enough free space for the whole file. EDLIN aborted the End command and returned you to the operating system. Some of the file may have been written to the disk.

Cure: Only a portion (if any) of the file has been saved. You should probably delete that portion of the file and restart the editing session. The file will not be available after this error. Always be sure that the disk has sufficient free space for the file to be written to disk <u>before</u> you begin your editing session.

Entry Error

Cause: The last command you typed contained an error.

Cure: Retype the command with the correct format and press the Return key.

File creation error

Cause: The EDLIN temporary file cannot be created.

Cure: Check to make sure that the directory has enough space to create the temporary file. Also, make sure that the file does not have the same name as a subdirectory in the directory where the file to be edited is located.

Filename must be specified

Cause: You did not type a filename when you started EDLIN.

Cure: Type EDLIN and then a filename.

File not found

Cause: EDLIN could not file the filename you typed with a Transfer command.

Cure: Type a valid filename when issuing a Transfer command.

Incorrect DOS version

Cause: You attempted to run EDLIN under a version of MS-DOS that was not 2.0 or higher.

Cure: You must make sure that the version of MS-DOS that you are using is 2.0 or higher.

Insufficient memory

Cause: There is not enough memory to run EDLIN.

Cure: You must free some memory by writing files to disk or by deleting files before restarting EDLIN. Type a Write command and then an Append command to free memory.

Invalid drive name or file

Cause: You did not type a valid drive or filename when you started EDLIN.

Cure: Type a correct drive or filename.

Invalid Parameter

Cause: You specified a switch other than /b when starting EDLIN.

Cure: Specify the /b switch when you start EDLIN.

Line too long

Cause: During a Replace command, the string given as the replacement caused the line to expand beyond the limit of 253 characters. EDLIN ended the Replace command.

Cure: Divide the long line into two lines, then try the Replace command again.

Must specify destination number

Cause: A destination line number was not specified for a Copy or Move command.

Cure: Retype the command with a destination line number.

No room in directory for file

Cause: When you tried to create a new file, either the directory was full or you specified an illegal disk drive or an illegal filename.

Cure: Check the command line that started EDLIN for an illegal filename or illegal disk drive entry. If the command is no longer on the screen and if you have not yet typed a new command, you can restart EDLIN by pressing the <COPYALL> key.

If your command contains no illegal entries, run the Chkdsk program for the specified disk drive. If the status report shows that the disk directory is full, remove the disk. Insert and format a new disk.

Not enough room to merge the entire file

Cause: There was not enough room in memory to hold the file during a Transfer command.

Cure: You must free some memory by writing some files to disk or by deleting some files before you can transfer this file.

Chapter 6
File Comparison Utility (FC)

CHAPTER 6

FILE COMPARISON UTILITY (FC)

6.1 INTRODUCTION

It is sometimes useful to compare files on your disk. If you have copied a file and later want to compare copies to see which one is current, you can use the MS-DOS File Comparison Utility (FC).

The File Comparison utility compares the contents of two files, or two sets of files. The differences between the two files can be sent to the screen or to a third file. The files being compared must be text files unless you specify the /b switch.

The comparisons are made in one of two ways:

1. on a line-by-line
2. or a byte-by-byte basis.

The line-by-line comparison isolates blocks of lines that are different between two files and prints those blocks of lines. You use this type of comparison to check for differences in text files. The byte-by-byte comparison displays the bytes that are different between two files. You use this type of comparison to check for differences in binary files.

6.2 LIMITATIONS ON COMPARISONS

FC uses a large amount of memory (enough to hold 100 lines) as buffer storage space to hold the text files. If the text files are larger than available memory, FC will compare what can be loaded into the buffer space. If no match is found in the portions of the files in the buffer space, FC stops and displays the message:

```
resynch failed.  Files are too different.
```

For binary files larger than available memory, FC compares both files completely, overlaying the portion in memory with the next portion from disk. All differences are output in the same manner as those files that fit completely in memory.

6.3 HOW TO USE FC

The syntax of FC is as follows:

```
     fc [/a] [/b] [/c] [/L] [/Lb <n>] [/n] [/t] [/w]
     [/<NNNN>] <filename1> <filename2>
```

The <filename1> options specifies the first file or set of files that you want to compare, while the <filename2> option specifies the second file or set of files that you want to compare. FC matches the first file against the second and reports any differences between them. For example, to compare two text files called MONTHLY.REP and SALES.REP you would type:

```
     fc /a monthly.rep sales.rep
```

If you want to specify a set of files, you use a wildcard as part of the filename. For example, if Joe and Sue both have files called FILE1.TXT, FILE2.TXT, and FILE3.TXT, they might use FC to see if their files are identical. They would use the command:

```
     fc b:\user\joe\file?.txt \user\sue\file?.txt
```

FC would take FILE1.TXT in the \USER\JOE directory of disk drive B and compares it with the corresponding FILE1.TXT in the \USER\SUE directory. Then FC would compare Joe's FILE2.TXT against Sue's FILE2.TXT, and then compare their FILE3.TXT files.

Since no drive was specified for the second set of files, FC would assume that the \USER\SUE directory was on the disk in the default drive.

6.4 FC SWITCHES

There are nine switches that you can use with the File Comparison Utility:

/a Abbreviates the output of an ASCII comparison. Instead of displaying all of the lines that are different, only the lines that begin and end each set of differences are displayed. The intermediate lines are represented by ellipses (...).

/b Forces a binary comparison of both files. The two files are compared byte-by-byte, with no attempt to re-synchronize after a mismatch. The mismatches are printed as follows:

```
xxxxxxxx   yy   zz
```

(where xxxxxxxx is the relative address of the pair of bytes from the beginning of the file). Addresses start at 00000000; yy and zz are the mismatched bytes from file1 and file2, respectively. If one of the files contains less data than the other, then a message is printed out. For example, if FILE1.TXT ends before FILE2.TXT, then FC displays:

```
fc: file2.txt longer than file1.txt
```

This switch is the default when you compare .EXE, .COM, .SYS, .OBJ, .LIB, OR .BIN files.

/c Causes the matching process to ignore the case of letters. All letters in the files are considered uppercase letters. For example,

```
Much MORE data IS NOT FOUND
```

matches:

```
much more data is not found
```

If both the /w and /c options are specified, then FC will compress whites and ignore case. For example, if an underscore represents a white:

```
___DATA_was_found____
```

matches:

```
data_was_found
```

/L Compares the files in ASCII mode. This switch is the default when you compare files that do not have extensions of ".EXE," ".COM," ".SYS," ".OBJ," ".LIB," OR ".BIN."

/Lb Sets the internal line buffer to <n> lines. The default length of the internal buffer is 100 lines. Files that have more than this number of consecutive, differing lines will abort the comparison.

/n Displays the line numbers on an ASCII comparison.

/t Does not expand tabs to spaces. The default is to treat tabs as spaces to 8-column positions.

/w Causes FC to compress whites (tabs and spaces) during the comparison. Thus, multiple contiguous whites in any line will be considered as a single white space. Note that although FC compresses whites, it does not ignore them. The two exceptions are beginning and ending whites in a line, which are ignored. For example (note that an underscore represents a white):

```
___More__data_to_be_found____
```

matches:

```
More_data_to_be_found
```

and:

```
_______More_______data_to_be_____found_______
```

but not:

```
____Moredata_to_be_found
```

/<NNNN> This switch specifies the number of lines required to match for the files to be considered as matching again after a difference has been found. If this switch is not specified, it defaults to 2.

6.5 HOW FC REPORTS DIFFERENCES

FC reports the differences between the two files you specify by displaying the first filename, followed by the lines that differ between the files, followed by the first line to match in both files. FC then displays the name of the second file followed by the lines that are different, followed by the first line that matches. The default for the number of lines to match between the files is 2. (If you want to change this default, specify the number of lines with the /<NNNN> switch.)

Example:

```
                ...
                ...

     ***** (filenamel)
     <lst matching line in filel and file2>
     <difference>
     <next matching line in filel and file2>
     ***** (filename2)
     <lst matching line in filel and file2>
     <difference>
     <next matching line in filel and file2>

                ...
                ...
```

FC will continue to list each difference.

If the number of lines in the internal buffer is less than the number of consecutive, differing lines, the program will stop. FC displays:

```
     resynch failed.  Files are too different.
```

and returns to the MS-DOS default drive prompt (for example, A>).

6.6 REDIRECTING FC OUTPUT TO A FILE

The differences and matches between the two files you specify will be displayed on your screen unless you redirect the output to a file. This is accomplished in the same way as MS-DOS command redirection (refer to Chapter 2, "Learning About Commands").

To compare FILE1.TXT and FILE2.TXT and then send the FC output to DIFFER.TXT, type:

```
     fc filel.txt file2.txt > differ.txt
```

The differences and matches between FILE1.TXT and FILE2.TXT will be put into DIFFER.TXT on the default drive.

6.7 EXAMPLES

Example 1:

Assume these two text files are on disk:

ALPHA.DOC	BETA.DOC
A	A
B	B
C	C
D	G
E	H
F	I
G	J
H	1
I	2
M	P
N	Q
O	R
P	S
Q	T
R	U
S	V
T	4
U	5
V	W
W	X
X	Y
Y	Z
Z	

To compare the two files and display the differences on the screen, type:

```
fc alpha.doc beta.doc
```

FC compares ALPHA.DOC with BETA.DOC and displays the differences on the screen. All other defaults remain intact. (The defaults are: perform an ASCII comparison with an internal buffer length of 100 lines, treat tabs as spaces to 8-column positions, do not ignore case, do not display line numbers, and do not compress whites.)

The output will appear as follows on the screen (the notes do not appear):

```
***** alpha.doc
C
D                       NOTE: ALPHA file
E                       contains CDEFG,
F                       BETA contains CG.
G
***** beta.doc
C
G
*****

***** alpha.doc
I
M                       NOTE: ALPHA file
N                       contains IMNO where
O                       BETA contains IJ12.
P
***** beta.doc
I
J
1
2
P
*****

***** alpha.doc
V                       NOTE: ALPHA file
W                       contains VW where
***** beta.doc          BETA contains V45W.
V
4
5
W
*****
```

Example 2:

You can print the differences on the line printer using the same two files. In this example, four successive lines must be the same to constitute a match.

Type:

```
     fc /4 alpha.doc beta.doc > prn
```

The following output would appear on the lineprinter:

```
     ***** alpha.doc
     C
     D
     E
     F
     G
     H
     I
     M
     N            NOTE: P is the 1st of
     O            a string of 4 matches.
     P
     ***** beta.doc
     C
     G
     H
     I
     J
     1
     2
     P
     *****

     ***** alpha.doc
     V
     W                        NOTE: W is the 1st of a
     ***** beta.doc           string of 4 matches.
     V
     4
     5
     W
     *****
```

Example 3:

This example forces a binary comparison and then displays the differences on the screen using the same two files as were used in the previous examples.

Type:

```
fc /b alpha.doc beta.doc
```

The /b switch in this example forces binary comparison. This switch and any others must be typed before the filenames in the FC command line. The following display would appear:

```
00000009   44   47
0000000C   45   48
0000000F   46   49
00000012   47   4A
00000015   48   31
00000018   49   32
0000001B   4D   50
0000001E   4E   51
00000021   4F   52
00000024   50   53
00000027   51   54
0000002A   52   55
0000002D   53   56
00000030   54   34
00000033   55   35
00000036   56   57
00000039   57   58
0000003C   58   59
0000003F   59   5A
00000042   5A   1A
fc: alpha.doc longer than beta.doc
```

6.8 ERROR MESSAGES

When FC detects an error, one or more of the following error messages will be displayed:

```
    fc:  cannot open (filename) - No such file or directory
         One of the files you specified does not exist.
         Check the directory for the correct filename.

    fc:  (filename) longer than (filename)
         After reaching the end of one of the files in a file
         comparison, the other file still has uncompared data
         left.

    fc:  incompatible switches
         You have specified switches that are not compatible.
         (For example, /b and /L.) You should not combine
         binary and ASCII comparison switches.

    fc:  out of memory
         You do not have enough memory to perform the
         comparison.

    fc:  no differences encountered
         The files are the same.

    usage:  fc [/a] [/b] [/c] [/l] [/lb n] [/w] [t] [/n]
               [/NNNN] file1 file2
         One of the switches that you have specified is
         invalid.

    resynch failed.  Files are too different.
         FC displays this message if the number of lines in
         the internal line buffer is less than the number of
         consecutive, differing lines.  You should specify
         the /Lb switch with a larger number if you want to
         display all of the file differences.
```

Chapter 7
LINK: A Linker

CHAPTER 7

LINK: A LINKER

7.1 INTRODUCTION

The Microsoft 8086 Object Linker (LINK) creates executable programs from object files generated by the Microsoft Macro Assembler (MASM) or by high-level-language compilers, such as C or Pascal. The linker copies the resulting program to an executable (.EXE) output file. You can then run the program by typing the file's name on the MS-DOS command line.

To use LINK, you must create one or more object files, then submit these files, along with any required library files, to the linker for processing. LINK combines code and data in the object files and searches the named libraries to resolve external references to routines and variables. It then copies a relocatable execution image and relocation information to the executable file. Using the relocation information, MS-DOS can load the executable image at any convenient memory location and execute it. LINK can process programs that contain up to one megabyte of code and data.

Section 7.2 explains how to use the linker to create executable programs. Section 7.3 defines each of the options you can use in a LINK command line to control the linking process. Section 7.4 explains how LINK creates programs.

7.2 STARTING AND USING LINK

This section explains how to start and use the linker to create executable programs. You can use LINK in three different ways: by answering a series of prompts, by supplying an MS-DOS command line, or by using a response file. The three methods can also be mixed.

Once you start LINK, it will either process the files you supplied or prompt you for additional files. You can stop the linker at any time by pressing the Control-C keys.

7.2.1 Using Prompts to Specify LINK Files

When you type the command name LINK at the MS-DOS prompt, the linker prompts you for the information it needs. Follow these steps:

1. Type:

   ```
   LINK
   ```

   and press the Return key. LINK prompts you for the object files you wish to link by displaying the following message:

   ```
   Object Modules [.OBJ]:
   ```

2. Type the name or names of the object files you wish to link. If you do not supply file-name extensions, LINK supplies .OBJ by default. If you have more than one name, make sure you separate them with spaces or plus signs (+). If you have more names than can fit on one line, type a plus sign (+) as the last character on the line and press the Return key. LINK prompts for additional object files.

   Once you have given all object-file names, press the Return key. The linker displays the following prompt:

   ```
   Run File [filename.EXE]:
   ```

3. Note that filename is the same as the first filename entered at the Object Modules prompt. Type the name of the executable file you wish to create, and press the Return key. If you do not give an extension, LINK supplies .EXE by default. If you want LINK to supply a default executable-file name, just press the Return key. The filename will be the same as the first object file, but the file will have the extension .EXE.

   Once you have pressed the Return key, LINK displays the prompt:

   ```
   List File [NUL.MAP]:
   ```

4. Type the name of the map file you wish to create, then press the Return key. If you do not supply a filename extension, the linker uses .MAP by default. If you do not want a map file, do not type a filename. Just press the Return key.

   Once you have pressed the Return key, LINK displays the prompt:

   ```
   Libraries [.LIB]:
   ```

5. Type the names of any library files containing routines or variables referenced but not defined in your program. If you give more than one name, make sure the names are separated by spaces or plus signs (+). If you do not supply filename extensions, the linker uses .LIB by default. If you have more names than can fit on one line, type a plus sign (+) as the last character on the line and press the Return key. LINK prompts for additional filenames.

   After entering all names, press the Return key. If you do not want to search any libraries, do not enter any names. Just press the Return key.

LINK now creates the executable file.

When entering file names, you must give a pathname for any file that is not on the current drive and directory. You can use LINK options by typing them after the filename at any prompt. If the linker cannot find an object file, it displays a message and waits so that you can change disks if necessary.

At any prompt, you can type the rest of the filenames in the command line format described in Section 7.2.2. For example, you can choose the default responses for all remaining prompts by typing a semicolon (;) after any prompt, or you can type commas (,) to indicate several files. (If you type a semicolon at the Object Modules prompt, be sure to supply at least one object-file name.) When the linker encounters a semicolon, it immediately chooses the default responses and processes the remaining files without displaying any more prompts.

7.2.1.1 Example -

```
LINK

Object Modules [.OBJ]: moda+modb+
Object Modules [.OBJ]: modc+startup/PAUSE
Run File [moda.EXE]:
List File [NUL.MAP]: abc
Libraries [.LIB]: b:\lib\math
```

This example links the object modules moda.obj, modb.obj, modc.obj, and startup.obj. It searches the library file math.lib on Drive B of the lib directory for routines and data used in the program. It then creates an executable file named moda.exe, and a map file named abc.map. The /PAUSE option in the Object Modules prompt line causes LINK to pause while you change disks. The linker then creates the executable file (see Section 7.3.2).

7.2.2 Using a Command Line to Specify LINK Files

You can create an executable program by typing LINK followed by the names of the files you wish to process. The command line has the following general form:

```
LINK <objectfiles> [,[<executablefile>] [,[<mapfile>]
[,[<libraryfile>]]]] [<options>] [;]
```

The <objectfiles> include the name or names of object files that you want to link together. The files must have been created using MASM or a high-level-language compiler. The linker requires at least one object file. If you do not supply an extension, LINK provides the extension .OBJ.

The optional <executablefile> is a placeholder for the name you wish to give the executable file LINK will create. If you do not supply an <executablefile>, LINK creates a filename by using the filename of the first object file in the command line and appending it with the extension .EXE.

The optional <mapfile> is the name of the file to receive the map listing. If you do not supply an extension, the linker provides the extension .MAP. If you specify the /MAP or /LINENUMBERS option, the linker creates a map file even if you do not specify a map file in the command line.

The optional <libraryfiles> include the name or names of the libraries containing routines that you wish to link to create a program. If you do not supply an extension, LINK supplies the extension .LIB.

The <options> control the operation of LINK. You can use any of the options listed in Section 7.3. You can put options anywhere on the command line.

The commas (,) separating filenames for the different types of files are required even if you don't supply a filename. If you want the filename for a file to be the default (the same as the base name of the first object file), you can type the comma that would follow the filename without actually supplying a filename. You can use a semicolon (;) anywhere after the object file to terminate the command line. If you type the comma after the object file, LINK supplies the default name for the <executablefile> and suppresses the <mapfile> and the <libraryfiles>.

If you do not supply all filenames in the command line and do not end the command line with a semicolon, the linker prompts for additional files, using the prompts described in Section 7.2.1. If you give more than one object file or library file, you must separate the names with spaces or with plus signs (+).

If you do not specify a drive or directory for a file, LINK assumes the file is on the current drive and directory. You cannot specify the drive or directory for the <objectfile> and expect LINK to supply the same drive and directory for other files. You must give the location of each file, specifically.

Note

When linking modules that were produced by a high-level-language compiler that supports overlays, you must specify overlay modules by putting them in parentheses. Since MASM has no overlay manager, you can only specify overlays for object files linked with the run-time library of a language compiler that supports overlays. For example, you can use overlays with modules compiled with Microsoft FORTRAN, Version 3.2 and later, Microsoft Pascal, Version 3.2 and later, and Microsoft C, Version 3.0 and later. See your language compiler manual for details on specifying overlays.

7.2.2.1 Examples -

```
    LINK file.obj,file.exe,file.map,routine.lib
```

The first example is equivalent to the following line:

```
    LINK file,,,routine
```

It uses the object file file.obj to create the executable file file.exe. LINK searches the library file.lib for routines and variables used within the program. It also creates a file called file.map, which contains a list of the program's segments and groups.

```
    LINK startup+file,b:file,\map\file;
```

The second example uses the two object files, startup.obj and file.obj, on the current drive to create an executable file named file.exe on Drive B. LINK creates a map file on the map directory of the current drive, but does not search any libraries.

```
    LINK moda modb modc startup/PAUSE,,abc,b:\lib\math
```

The final example links the object modules moda.obj, modb.obj, modc.obj, and startup.obj. The linker searches through the library file math.lib in the lib directory on Drive B for routines and data used in the program. It then creates an executable file named moda.exe, and a map file named abc.map. The PAUSE option in the command line causes the linker to pause while you change disks before creating the executable file (see Section 7.3.2).

7.2.3 Using a Response File to Specify LINK Files

You can create a program by listing, in a response file, the names of all the files to be processed, and by giving the name of the response file on the LINK command line. The simplest way to use a response file is with a command line having the following form:

```
    LINK \@<filename>
```

You can also specify a response file at any prompt, or at any position in a command line. The input from the response file is treated exactly as though it had been entered at prompts or in a command line, except that carriage-return/linefeed combinations in the file are treated the same as the Return key in response to a prompt, or a comma in a command line.

When you specify a response file, remember that the filename must be the name of the response file, and that you must precede it by an at sign (@). If the file is in another directory or on another disk drive, you must provide a path name.

You can name the response file anything you like. The file content has the following general form:

```
<objectfiles>
[<executablefile>]
[<mapfile>]
[<libraryfiles>]
```

You can omit any elements that have already been provided at prompts or with a partial command line.

You must place each group of filenames on a separate line. If you have more names than can fit on one line, you can continue the names on the next line by typing a plus sign (+) as the last character in the current line. If you do not supply a filename for a group, you must leave an empty line. You can give options on any line.

You can place a semicolon (;) on any line in the response file. When LINK encounters the semicolon, it automatically supplies default file names for all files you have not yet named in the response file. The remainder of the response file is ignored.

When you create a program with a response file, the linker displays each response from your response file on the screen in the form of prompts. If the response file does not contain names for required files, LINK prompts for the missing names and waits for you to enter responses.

Note

A response file should end with either a semicolon (;) or a carriage-return/line-feed combination. If you fail to provide a final carriage-return/linefeed in the file, the linker will display the last line of the response file and wait for you to press the Return key.

7.2.3.1 Example -

```
moda modb modc startup /PAUSE
abc
b:\lib\math
```

The response file above tells the linker to link the four object modules moda, modb, modc, and startup. LINK pauses to let you swap disks before producing the executable file moda.exe. The linker also creates a map file, abc.map, and searches the library, math.lib, in the \lib directory of Drive B.

The following procedure combines all three methods of supplying filenames. Assume you have a response file called library that contains one line:

```
lib1+lib2+lib3+lib4
```

Now start LINK with a partial command line:

```
LINK object1 object2
```

LINK takes object1.obj and object2.obj as its object files, and prompts for the next file:

```
Run File [object1.EXE]: exec
List File [NUL.MAP]:
Libraries [.LIB]: @library
```

You enter exec so that the linker will name the executable file exec.exe. You then press the Return key to indicate that no map file is desired, and you enter @library so that the linker will read in the response file containing the four library-filenames.

7.2.4 Giving Search Paths with Libraries

You can direct LINK to search directories and disk drives for the libraries you have named in a command by specifying one or more search paths with the library names, or by assigning the search paths to the environment variable LIB before you invoke LINK. Environment variables are explained under the SET command in Chapter 3, "MS-DOS Commands."

A search path is the path specification of a directory or drive name. You type search paths along with library names on the LINK command line or in response to the Libraries prompt. You can specify up to 16 search paths. You can also assign the search paths to the LIB environment variable, using the MS-DOS SET command. In the latter case, you must separate the search paths by semicolons (;).

If you include a drive or directory name in the filename for a library in the LINK command line, the linker searches there only. But if you don't give a drive or directory name, LINK searches for library files in the following order:

1. First, the linker searches the current drive and directory.

2. If the library is not found and one or more search paths have been given in the command line, the linker searches the specified search paths in the order in which you gave them.

3. If the library is still not found and you have set a search path by using the LIB environment variable, the linker searches there.

4. If the library is still not found, LINK prints an error message.

7.2.4.1 Examples -

```
     LINK file,,file,A:\altlib\math.lib+common+B:+D:\lib\
```

In the first example, the linker searches only the \altlib directory on drive A to find the library, math.lib, but to find common.lib it will search the current directory on the current drive, the current directory on drive B, and finally, the directory \lib on drive D.

```
     SET LIB=C:\lib;U:\system\lib
     LINK file,,file.map,math+common
```

In the second example, LINK searches the current directory, the directory \lib on drive C, and the directory systemlib on drive U to find the libraries, math.lib and common.lib.

7.2.5 The Map File

The map file lists the names, load addresses, and lengths of all segments in a program. It also lists the names and load addresses of any groups in the program, the program start address, and messages about any errors it may have encountered. If the /MAP option is used in the LINK command line, the map file lists the names and load addresses of all public symbols.

Segment information has the general form shown in this example:

```
Start   Stop    Length  Name            Class

00000H  0172CH  0172DH  TEXT            CODE
01730H  01E19H  006EAH  DATA            DATA
```

The Start and Stop columns show the 20-bit addresses (in hexadecimal) of the first and last byte in each segment. These addresses are relative to the beginning of the load module, which is assumed to be address 0000H. The operating system chooses its own starting address when the program is actually loaded. The Length column gives the length of the segment in bytes; the Name column, the name of the segment; and the Class column, the segment's class name.

Group information has the general form:

```
Origin  Group

0000:0  IGROUP
0173:0  DGROUP
```

In this example, IGROUP is the name of the code (instruction) group and DGROUP is the name of the data group.

At the end of the listing file, the linker gives you the address of the program entry point.

If you specify the /MAP option in the LINK command line, the linker adds a public-symbol list to the map file. The symbols are presented twice: once in alphabetical order, then in the order of their load addresses. The list has the general form shown in the following example:

```
 Address         Publics by Name

0000:1567       BRK
0000:1696       CHMOD
0000:01DB       CHKSTK
0000:131C       CLEARERR
0173:0035       FAC

 Address         Publics by Value

0000:01DB       CHKSTK
0000:131C       CLEARERR
0000:1567       BRK
0000:1696       CHMOD
0000:0035       FAC
```

The addresses of the public symbols are in segment:offset format. They show the location of the symbol relative to the beginning of the load module, which is assumed to be at address 0000:0000.

When the /HIGH and /DSALLOCATE options are used (see Sections 7.3.10 and 7.3.11) and the program's code and data combined do not exceed 64K, the map file may show symbols that have unusually large segment addresses. These addresses indicate a symbol whose location is below the actual start of the program code and data. For example, the symbol entry:

```
FFF0:0A20       TEMPLATE
```

shows that TEMPLATE is located below the start of the program. Note that the 20-bit address of TEMPLATE is 00920h.

7.2.6 The Temporary Disk File - VM.TMP

LINK normally uses available memory for the link session. If it runs out of available memory, it creates a temporary disk file named VM.TMP in the current working directory. When the linker creates this file, it displays the following message:

```
VM.TMP has been created.
Do not change diskette in drive letter
```

Note that letter will be the proper drive name. After this message appears, you must not remove the disk from the drive specified by letter until the link session ends. The /PAUSE option cannot be used if a temporary file is created. After LINK has created the executable file, it deletes the temporary file automatically.

Note

Do not use the filename, VM.TMP, for your own files. When the linker creates the temporary file, it destroys any previous file that have the same name.

7.3 USING LINK OPTIONS

The linker options specify and control the tasks that LINK performs. All options begin with the linker-option character, which is the forward slash (/). You can use an option anywhere on a LINK command line.

LINK has the following options:

Option Name	Action
/HELP	Shows options list
/PAUSE	Pauses during linking
/EXEPACK	Packs executable file
/MAP	Creates public symbol map
/LINENUMBERS	Copies line numbers to map file
/NOIGNORECASE	Preserves case sensitivity in names
/NODEFAULTLIBRARYSEARCH	Overrides default libraries
/STACK	Sets maximum allocation space
/HIGH	Sets high load address
/DSALLOCATE	Allocates data group
/NOGROUPASSOCIATION	Sets group association override
/OVERLAYINTERRUPT	Sets overlay interrupt
/SEGMENTS	Sets maximum number of segments
/DOSSEG	Specifies MS-DOS segment ordering

You can abbreviate option names as long as your abbreviations contain enough letters to distinguish the specified option from other options. Minimum abbreviations are listed for each option.

Many of the LINK options set values in the MS-DOS program header. You will understand these options better if you understand how the header is organized. The program header is described in the MS-DOS Programmer's Reference and in some reference books on MS-DOS.

7.3.1 Viewing the Options List

7.3.1.1 Syntax -

/HELP

The /HELP option causes LINK to write a list of the available options to the screen. This may be convenient if you need a reminder of the available options. You should not give a filename when using the /HELP option.

Minimum abbreviation: /HE

7.3.1.2 Example -

```
    LINK /HELP
```

7.3.2 Pausing to Change Disks

7.3.2.1 Syntax -

/PAUSE

The /PAUSE option causes LINK to pause before writing the executable file to disk so that you can swap disks before the linker writes the executable (.EXE) file to disk.

If you specify the /PAUSE switch, the linker displays the following message before creating the run file:

```
About to generate .EXE file
Change diskette in drive letter and press <ENTER>
```

Note that letter is the proper drive name. This message appears after the linker has read data from the object files and library files, and after it has written data to the map file, if you specified one. LINK resumes processing when you press the Return key. After LINK writes the executable file to disk, the following message appears:

```
Please replace original diskette
in drive letter and press <ENTER>
```

Minimum abbreviation: /P

Note

Do not remove the disk used for the VM.TMP file, if such a file has been created. If the temporary disk message appears when you have specified the /PAUSE option, you should press Control-C to terminate the LINK session. Rearrange your files so that the temporary file and the executable file can be written to the same disk, then try again.

7.3.2.2 Example -

```
    LINK file/PAUSE,file,,\lib\math
```

This command causes the linker to pause just before creating the executable file, file.exe. After creating the executable file, LINK pauses again to let you replace the original disk.

7.3.3 Packing Executable Files

7.3.3.1 Syntax -

/EXEPACK

The /EXEPACK option directs LINK to remove sequences of repeated bytes (typically nulls) and optimize the load-time relocation table before creating the executable file. Executable files linked with the option may be smaller, and thus load faster than files linked without the option. However, the Microsoft Symbolic Debug Utility (SYMDEB) cannot be used with packed files.

The /EXEPACK option does not always save a significant amount of disk space (and may sometimes actually increase file size). Programs that have a large number of load-time relocations (about 500 or more) and long streams of repeated characters will usually be shorter if packed. If you're not sure if your program meets these conditions, try linking it both ways and compare the results.

Minimum abbreviation: /E

7.3.3.2 Example -

```
     LINK program /E ;
```

This example creates a packed version of file program.exe.

7.3.4 Producing a Public-Symbol Map

7.3.4.1 Syntax -

/MAP

The /MAP option causes LINK to produce a listing of all public symbols declared in your program. This list is copied to the map file that LINK creates. For a complete description of the listing-file format, see Section 7.2.5. The /MAP option is required if you want to use SYMDEB for symbolic debugging.

Minimum abbreviation: /M

Note

If you do not specify a map file in a LINK command, you can use the /MAP option to force the linker to create a map file. LINK gives the forced map file the same filename as the first object file specified in the command. It also adds the default extension .MAP.

7.3.4.2 Example -

```
LINK file,,/MAP;
```

This command creates a map of all public symbols in the file file.obj.

7.3.5 Copying Line Numbers to the Map File

7.3.5.1 Syntax -

/LINENUMBERS

The /LINENUMBERS option directs the linker to copy the starting address of each program source line to a map file. The starting address is actually the address of the first instruction that corresponds to the source line. You can use the MAPSYM program to copy line-number data to a symbol file, which can then by used by SYMDEB.

The linker copies the line-number data only if you give a map-file name in the LINK command line, and only if the given object file has line-number information. Line numbering is available in some high-level-language compilers, including Microsoft FORTRAN and Pascal, versions 3.0 and later, and Microsoft C, version 2.0 and later.

MASM does not copy line-number information to the object file. If an object file has no line-number information, the linker ignores the /LINENUMBERS option.

Minimum abbreviation: /LI

Note

If you do not specify a map file in a LINK command, you can still use the /LINENUMBERS option to force the linker to create a map file. Just place the option at or before the List File prompt. LINK gives the forced map file the same filename as the first object file that you specified in the command and gives it the default extension, .MAP.

7.3.5.2 Example -

```
    LINK file/LINENUMBERS,,em+slibfp
```

This example causes the line-number information in the object file file.obj to be copied to the map file file.map.

7.3.6 Preserving Lowercase

7.3.6.1 Syntax -

/NOIGNORECASE

The /NOIGNORECASE option directs LINK to treat uppercase and lowercase letters in symbol names as distinct letters. Normally, LINK considers uppercase and lowercase letters to be identical, treating the names TWO, two, and Two as the same symbol. When you use the /NOIGNORECASE option, the linker treats TWO, Two, and two as different symbols.

Typically, you use the /NOIGNORECASE option with object files created by high-level-language compilers. Some compilers treat uppercase and lowercase letters as distinct letters and assume the linker does the same.

If you are linking modules, created with MASM, to modules created with a case-sensitive language such as C, make sure public symbols have the same sensitivity in both modules. For example, you could make all variables in C distinctive by spelling, regardless of case, and then link without the /NOIGNORECASE option. Another alternative would be to use the /ML or MX option to make public variables in MASM case-sensitive. Then link with the /NOIGNORECASE option.

Minimum abbreviation: /NOI

7.3.6.2 Example -

```
    LINK file1+file2/NOI,,,em+mlibfp
```

This command causes the linker to treat uppercase and lowercase letters in symbol names as distinct letters. The object file, file.obj, is linked with routines from the standard C language library, \Slibc.lib, located in the \lib directory. The C language expects uppercase and lowercase letters to be treated as distinct.

7.3.7 Ignoring Default Libraries

7.3.7.1 Syntax -

/NODEFAULTLIBRARYSEARCH

The /NODEFAULTLIBRARYSEARCH option directs the linker to ignore any library names it may find in an object file. A high-level-language compiler may add a library name to an object file to ensure that a default set of libraries is linked with the program. Using this option overrides these default libraries and lets you explicitly name the libraries you want by including them on the LINK command line.

Minimum abbreviation: /NOD

7.3.7.2 Example -

```
     LINK startup+file/NOD,,,em+slibfp+slibc
```

This example links the object files, startup.obj and file.obj with routines from the libraries em, slibfp, and slibc. Any default libraries that may have been named in startup.obj or file.obj are ignored.

7.3.8 Setting the Stack Size

7.3.8.1 Syntax -

/STACK:<size>

The /STACK option sets the program stack to the number of bytes given by size. The linker usually calculates a program's stack size automatically, basing the size on the size of any stack segments given in the object files. If /STACK is given, the linker uses the given size in place of any value it may have calculated.

The size can be any positive integer value in the range 1 to 65535. The value can be a decimal, octal, or hexadecimal number. Octal numbers must begin with a zero. Hexadecimal numbers must begin with a leading zero followed by a lowercase x. For example, Ox1B.

By using the EXEMOD utility, you can also change the stack size after linking.

Minimum abbreviation: /ST

7.3.8.2 Examples -

```
     LINK file/STACK:512,,;
```

The first example sets the stack size to 512 bytes.

```
     LINK moda+modb,run/ST:0xFF,ab,\lib\start;
```

The second example sets the stack size to 255 (FFh) bytes.

```
     LINK startup+file/ST:030,,;
```

The final example sets the stack size to 24 (30 octal) bytes.

7.3.9 Setting the Maximum Allocation Space

7.3.9.1 Syntax -

/CPARMAXALLOC:<number>

The /CPARMAXALLOC option sets the maximum number of 16-byte paragraphs needed by a program when it is loaded into memory. The operating system uses this number when allocating space for a program prior to loading it.

LINK normally sets the maximum number of paragraphs to 65535. Since this represents all addressable memory, the operating system always denies the request and allocates the largest contiguous block of memory it can find. If you use the /CPARMAXALLOC option, the operating system allocates no more space than given by this option. This means any additional space in memory is free for other programs.

The <number> can be any integer value in the range 1 to 65535. It must be a decimal, octal, or hexadecimal number. Octal numbers must begin with a zero. Hexadecimal values must begin with a leading zero followed by a lowercase x. For example, 0x2B.

If <number> is less than the minimum number of paragraphs needed by the program, LINK ignores your request and sets the maximum value equal to the minimum needed. The minimum number of paragraphs needed by a program is never less than the number of paragraphs of code and data in the program.

Minimum abbreviation: /C

7.3.9.2 Examples -

```
     LINK file/C:15,,;
```

The first example sets the maximum allocation to 15 paragraphs.

```
     LINK moda+modb,run/CPARMAXALLOC:0xff,ab;
```

The second example sets the maximum allocation to 255 (FFh) paragraphs.

```
     LINK startup+file,/C:030,;
```

The final example sets the maximum allocation to 24 (30 octal) paragraphs.

7.3.10 Setting a High Start Address

7.3.10.1 Syntax -

/HIGH

The /HIGH option sets a program's starting address to the highest possible address in free memory. If you don't use the /HIGH option, LINK sets the program's starting address as low as possible in memory.

Minimum abbreviation: /H

7.3.10.2 Example -

```
     LINK startup+file/HIGH,,;
```

This example sets the starting address of the program in file.exe to the highest possible address in free memory.

7.3.11 Allocating a Data Group

7.3.11.1 Syntax -

/DSALLOCATE

The /DSALLOCATE option directs the linker to reverse its normal processing when assigning addresses to items belonging to the group named DGROUP. Normally, LINK assigns the offset 0000h to the lowest byte in a group. If you use /DSALLOCATE, LINK assigns the offset FFFFh to the highest byte in the group. The result is data that appear to be loaded as high as possible in the memory segment containing DGROUP.

Typically, you use the /DSALLOCATE option with the /HIGH option to take advantage of unused memory before the start of the program. The linker assumes that all free bytes in DGROUP occupy the memory preceding the program. To use the group, you must set a segment register to the start address of DGROUP.

Minimum abbreviation: /D

7.3.11.2 Example -

```
    LINK startup+file/HIGH/DSALLOCATE,,,em+mlibfp
```

This example directs the linker to place the program as high in memory as possible, then adjust the offsets of all data items in DGROUP so that they are loaded as high as possible within the group.

7.3.12 Removing Groups From a Program

7.3.12.1 Syntax -

/NOGROUPASSOCIATION

The /NOGROUPASSOCIATION option directs LINK to ignore group associations when assigning addresses to data and code items.

Minimum abbreviation: /NOG

Note

This option exists strictly for compatibility with older versions of FORTRAN and Pascal (Microsoft version 3.13 or earlier, or any IBM version prior to 2.0). You should never use the /NOGROUPASSOCIATION option except to link with object files produced by those compilers, or with the run-time libraries that accompany the old compilers.

7.3.13 Setting the Overlay Interrupt

7.3.13.1 Syntax -

```
/OVERLAYINTERRUPT:<number>
```

The /OVERLAYINTERRUPT option sets the interrupt number of the overlay loading routine to number. This option overrides the normal overlay interrupt number (03Fh).

The number can be any integer value in the range 0 to 255. It must be a decimal, octal, or hexadecimal number. Octal numbers must have a leading zero. Hexadecimal numbers must start with a leading zero followed by a lowercase x. For example, 0x3B.

MASM does not have an overlay manager. Therefore, you can only use this option if you are linking with a run-time module from a language compiler that supports overlays. Check your compiler documentation, since this option is not appropriate for use with some compilers.

Minimum abbreviation: /O

Note

You should not use interrupt numbers that conflict with the standard MS-DOS interrupts.

7.3.13.2 Examples -

```
     LINK file/O:255,,,87+slibfp
```

The first example sets the overlay interrupt number to 255.

```
     LINK moda+modb, run/OVERLAY:0xff,ab.map,em+mlibfp
```

The second example sets the overlay interrupt number to 255 (FFh).

```
     LINK startup+file,/O:0377,,em+mlibfp
```

The final example sets the overlay interrupt number to 255 (377 octal).

7.3.14 Setting the Maximum Number of Segments

7.3.14.1 Syntax -

/SEGMENTS:<number>

The /SEGMENTS option directs the linker to process no more than <number> segments per program. If it encounters more than the given limit, the linker displays an error message, and stops linking. You use this option to override the default limit of 128 segments.

If you do not use /SEGMENTS, the linker allocates enough memory space to process up to 128 segments. If your program has more than 128 segments, you will need to set the segment limit higher to increase the number of segments that LINK can process. If you get the following LINK error message:

```
Segment limit set too high
```

you should set the segment limit lower.

The number can be any integer value in the range 1 to 1024. It must be a decimal, octal, or hexadecimal number. Octal numbers must have a leading zero. Hexadecimal numbers must start with a leading zero followed by a lowercase x. For example, 0x4B.

Minimum abbreviation: /SE

7.3.14.2 Examples -

```
    LINK file/SE:192,,;
```

The first example sets the segment limit to 192.

```
    LINK moda+modb,run/SEGMENTS:0xff,ab,em+mlibfp;
```

The second example sets the segment limit to 255 (FFh).

7.3.15 Using DOS Segment Order

7.3.15.1 Syntax -

/DOSSEG

The /DOSSEG option causes LINK to arrange all segments in the executable file according to the MS-DOS segment-ordering convention. This convention has the following rules:

1. All segments having the class name CODE are placed at the beginning of the executable file.

2. Any other segments that do not belong to the group named DGROUP are placed immediately after the CODE segments.

3. All segments belonging to DGROUP are placed at the end of the file.

If you do not use the /DOSSEG option, see Section 7.4.3 for an explanation of the normal segment order.

Minimum abbreviation: /DO

7.3.15.2 Example -

```
    LINK start+test/DOSSEG,,math+common
```

This command causes the linker to create an executable file, named file.exe, whose segments are arranged according to the MS-DOS segment-ordering convention. The segments in the object files start.obj and test.obj, and any segments copied from the libraries, math.lib and common.lib, are arranged in the order specified above.

7.4 HOW LINK WORKS

LINK creates an executable file by concatenating a program's code and data segments according to the instructions in the original source files. These concatenated segments form an "executable image," that is copied directly into memory when you invoke the program for execution. Thus, the order and manner in which the linker copies segments to the executable file defines the order and manner in which it loads the segments into memory.

You can tell the linker how to link a program's segments by using a SEGMENT directive to supply segment attributes or by using the GROUP directive to form segment groups. These directives define group associations, classes, and align and combine types that define the order and relative starting addresses of all segments in a program. This information works in addition to any information you supply through command-line options.

The following sections explain the process that LINK uses to concatenate segments and resolve references to items in memory.

7.4.1 Alignment of Segments

The linker uses a segment's align type to set the starting address for the segment. The align types are byte, word, para, and page. These types correspond to starting addresses at byte, word, paragraph, and page boundaries, representing addresses that are multiples of 1, 2, 16, and 256, respectively. The default align type is para.

When the linker encounters a segment, it checks the align type before copying the segment to the executable file. If the align type is word, para, or page, the linker checks the executable image to see if the last byte copied ends at an appropriate boundary. If not, LINK pads the image with extra null bytes.

7.4.2 Frame Number

The linker computes a starting address for each segment in a program. The starting address is based on a segment's align type and the size of the segments already copied to the executable file. The address consists of an offset and a canonical frame number, which specifies the address of the first paragraph in memory that contains one or more bytes of the segment. A frame number is always a multiple of 16 (a paragraph address). But the offset is the number of bytes from the start of the paragraph to the first byte in the segment. For byte and word align types, the offset may be nonzero. The offset is always zero for para and page align types.

The frame number of a segment can be obtained from a LINK file. The frame number is the first five hexadecimal digits of the start address specified for the segment.

7.4.3 Order of Segments

LINK copies segments to the executable file in the same order that it encounters them in the object files. The linker maintains this order throughout the program unless it encounters two or more segments with the same class name. Segments with identical class names belong to the same class type, and are copied to the executable file as contiguous blocks.

For a more detailed discussion of segment loading order and methods of controlling loading order by assigning class types, see the Microsoft Macro Assembler Reference Manual.

7.4.4 Combined Segments

LINK uses combine types to determine whether two or more segments sharing the same segment name should be combined into a single, large segment. The combine types are public, stack, common, memory, at, and private. Combine types are also described in the Microsoft Macro Assembler Reference Manual.

If a segment has combine type public, the linker automatically combines it with any other segments that have the same name and belong to the same class. When LINK combines segments, it ensures that the segments are contiguous and that all addresses in the segments can be accessed using an offset from the same frame address. The result is the same as if the segment were defined as a whole in the source file.

The linker preserves each segment's align type. This means that even though the segments belong to a single, large segment, the code and data in the segments retain their original align type. If the combined segments exceed 64K, LINK displays an error message.

If a segment has combine type stack, the linker carries out the same combine operation as for public segments. The only difference is that stack segments cause LINK to copy an initial stack-pointer value to the executable file. This stack-pointer value is the offset to the end of the first stack segment (or combined stack segment) that the linker encounters. If you use the stack type for stack segments, you do not need to give instructions that load the segment into the SS register.

If a segment has combine type common, the linker combines it automatically with any other segments that have the same name and belong to the same class. When LINK combines common segments, however, it places the start of each segment at the same address, creating a series of overlapping segments. The result is a single segment which is no larger than the largest of the combined segments.

The linker treats segments with combine type memory exactly like segments with combine type public. MASM provides combine type memory for compatibility with linkers that support a separate combine type for memory segments.

A segment has combine type private only if no explicit combine type is defined for it in the source file. LINK does not combine private segments.

7.4.5 Groups

Groups permit non-contiguous segments that do not belong to the same class to be addressable relative to the same frame address. When LINK encounters a group, it adjusts all memory references to items in the group so that they are relative to the same frame address.

Segments in a group do not have to be contiguous, do not have to belong to the same class, and do not have to have the same combine type. The only requirement is that all segments in the group fit within 64K.

Groups do not affect the order in which the segments are loaded. Unless you use class names and enter object files in the right order, there is no guarantee that the segments will be contiguous. In fact, the linker may place segments that do not belong to the group in the same 64K of memory. Although LINK does not explicitly check that all segments in a group fit within 64K of memory, the linker is likely to encounter a "fixup-overflow" error if this requirement is not met.

Groups, and how to define them, are discussed in the Microsoft Macro Assembler Reference Manual.

7.4.6 Fixups

Once the starting address of each segment in a program is known, and all segment combinations and groups have been established, the linker can fix up any unresolved references to labels and variables. To fix up unresolved references, the linker computes an appropriate offset and segment address and replaces the temporary values, generated by the assembler, with the new values.

LINK carries out fixups for four different references:

- Short
- Near self-relative
- Near segment-relative
- Long

The size of the value to be computed depends on the type of reference. If LINK discovers an error in the anticipated size of a reference, it displays a fixup-overflow message. This can happen, for example, if a program attempts to use a 16-bit offset to reach an instruction in a segment that has a different frame address. It can also occur if all segments in a group do not fit within a single 64K block of memory.

A short reference occurs in JMP instructions that attempt to pass control to labeled instructions that are in the same segment or group. The target instruction must be no more than 128 bytes from the point of reference. The linker computes a signed, 8-bit number for this reference and displays an error message if the target instruction belongs to a different segment or group (has a different frame address), or if the target is more than 128 bytes distant (in either direction).

A near self-relative reference occurs in instructions which access data relative to the same segment or group. The linker computes a 16-bit offset for this reference and displays an error message if the data are not in the same segment or group.

A near segment-relative reference occurs in instructions which attempt to access data that is in a specified segment or group, or that is relative to a specified segment register. LINK computes a 16-bit offset for this reference and displays an error message if the offset of the target within the specified frame is greater than 64K or less than 0, or if the beginning of the canonical frame of the target is not addressable.

A long reference occurs in CALL instructions that attempt to access an instruction in another segment or group. LINK computes a 16-bit frame address and 16-bit offset for this reference. The linker displays an error message if the computed offset is greater than 64K or less than 0, or if the beginning of the canonical frame of the target is not addressable.

Chapter 8
DEBUG

CHAPTER 8

DEBUG

8.1 INTRODUCTION

The Microsoft DEBUG Utility (DEBUG) is a debugging program that provides a controlled testing environment for binary and executable object files. Note that EDLIN is used to alter source files; DEBUG is EDLIN's counterpart for binary files. DEBUG eliminates the need to reassemble a program to see if a problem has been fixed by a minor change. It allows you to alter the contents of a file or the contents of a CPU register, and then to immediately reexecute a program to check on the validity of the changes.

All DEBUG commands may be aborted at any time by pressing Control-C. Control-S suspends the display, so that you can read it before the output scrolls away. Entering any key other than Control-C or Control-S restarts the display. All of these commands are consistent with the control character functions available at the MS-DOS command level.

8.2 HOW TO START DEBUG

DEBUG may be started two ways. By the first method, you type all commands in response to the DEBUG prompt (a hyphen). By the second method, you type all commands on the line used to start DEBUG.

Summary of Methods to Start DEBUG

Method 1	DEBUG
Method 2	DEBUG [<filespec> [<arglist>]]

8.2.1 Method 1: DEBUG

To start DEBUG using method 1, type:

```
    DEBUG
```

DEBUG responds with the hyphen (-) prompt, signaling that it is ready to accept your commands. Since no filename has been specified, current memory, disk sectors, or disk files can be worked on by using other commands.

Warnings

1. When DEBUG (Version 2.0) is started, it sets up a program header at offset 0 in the program work area. On previous versions of DEBUG, you could overwrite this header. You can still overwrite the default header if no <filespec> is given to DEBUG. If you are debugging a .COM or .EXE file, however, do not tamper with the program header below address 5CH, or DEBUG will terminate.

2. Do not restart a program after the "Program terminated normally" message is displayed. You must reload the program with the N and L commands for it to run properly.

8.2.2 Method 2: Command Line

To start DEBUG using a command line, type:

```
    DEBUG [<filespec> [<arglist>]
```

For example, if a <filespec> is specified, then the following is a typical command to start DEBUG:

```
    DEBUG FILE.EXE
```

DEBUG then loads FILE.EXE into memory starting at 100 hexadecimal in the lowest available segment. The BX:CX registers are loaded with the number of bytes placed into memory.

An <arglist> may be specified if <filespec> is present. The <arglist> is a list of filename parameters and switches that are to be passed to the program <filespec>. Thus, when <filespec> is loaded into memory, it is loaded as if it had been started with the command:

```
     <filespec> <arglist>
```

Here, <filespec> is the file to be debugged, and the <arglist> is the rest of the command line that is used when <filespec> is invoked and loaded into memory.

8.3 COMMAND INFORMATION

Each DEBUG command consists of a single letter followed by one or more parameters. Additionally, the control characters and the special editing functions described in the MS-DOS User's Guide, apply inside DEBUG.

If a syntax error occurs in a DEBUG command, DEBUG reprints the command line and indicates the error with an up-arrow (^) and the word "error."

For example:

```
     dcs:100 cs:110
         ^ error
```

Any combination of uppercase and lowercase letters may be used in commands and parameters.

The DEBUG commands are summarized in Table 8.1 and are described in detail, with examples, following the description of command parameters.

Table 8.1 DEBUG Commands

DEBUG Command	Function
A[<address>]	Assemble
C<range> <address>	Compare
D[<range>]	Dump
E<address> [<list>]	Enter
F<range> <list>	Fill
G[=<address> [<address>...]]	Go
H<value> <value>	Hex
I<value>	Input
L[<address> [<drive:><record><record>]]	Load
M<range> <address>	Move
N<filename>[<filename>]	Name
O<value> <byte>	Output
Q	Quit
R[<register-name>]	Register
S<range> <list>	Search
T[=<address>][<value>]	Trace
U[<range>]	Unassemble
W[<address> [<drive:><record><record>]]	Write

8.4 PARAMETERS

All DEBUG commands accept parameters, except the Quit command. Parameters may be separated by delimiters (spaces or commas), but a delimiter is required only between two consecutive hexadecimal values. Thus, the following commands are equivalent:

```
dcs:100 110
d cs:100 110
d,cs:100,110
```

PARAMETER	DEFINITION
<drive:>	A one-digit hexadecimal value to indicate which drive a file will be loaded from or written to. The valid values are 0-3. These values designate the drives as follows: 0=A:, 1=B:, 2=C:, 3=D:.
<byte>	A two-digit hexadecimal value to be placed in or read from an address or register.
<record>	A 1- to 3-digit hexadecimal value used to indicate the logical record number on the disk and the number of disk sectors to be written or loaded. Logical records correspond to sectors. However, their numbering differs since they represent the entire disk space.
<value>	A hexadecimal value up to four digits used to specify a port number or the number of times a command should repeat its functions.
<address>	A two-part designation consisting of either an alphabetic segment register designation or a four-digit segment address plus an offset value. The segment designation or segment address may be omitted, in which case the default segment is used. DS is the default segment for all commands except G, L, T, U, and W, for which the default segment is CS. All numeric values are hexadecimal.

For example:

```
CS:0100
04BA:0100
```

The colon is required between a segment designation (whether numeric or alphabetic) and an offset.

<range> Two <address>es: e.g., <address> <address>; or one <address>, an L, and a <value>: e.g., <address> L <value> where <value> is the number of lines the command should operate on, and L80 is assumed. The last form cannot be used if another hex value follows the <range>, since the hex value would be interpreted as the second <address> of the <range>.

Examples:

```
CS:100 110
CS:100 L 10
CS:100
```

The following is illegal:

```
CS:100 CS:110
       ^ error
```

The limit for <range> is 10000 hex. To specify a <value> of 10000 hex within four digits, type 0000 (or 0).

<list> A series of <byte> values or of <string>s. <list> must be the last parameter on the command line.

Example:

```
fcs:100 42 45 52 54 41
```

<string> Any number of characters enclosed in quote marks. Quote marks may be either single (') or double("). If the delimiter quote marks must appear within a <string>, the quote marks must be doubled. For example, the following strings are legal:

```
'This is a "string" is okay.'
'This is a ''string'' is okay.'
```

However, this string is illegal:

```
'This is a 'string' is not.'
```

Similarly, these strings are legal:

```
"This is a 'string' is okay."
"This is a ""string"" is okay."
```

However, this string is illegal:

    "This is a "string" is not."

Note that the double quote marks are not necessary in the following strings:

    "This is a ''string'' is not necessary."
    'This is a ""string"" is not necessary.'

The ASCII values of the characters in the string are used as a <list> of byte values.

NAME

Assemble

PURPOSE

Assembles 8086/8087/8088 mnemonics directly into memory.

SYNTAX

A[<address>]

COMMENTS

If a syntax error is found, DEBUG responds with

```
^Error
```

and redisplays the current assembly address.

All numeric values are hexadecimal and must be entered as 1-4 characters. Prefix mnemonics must be specified in front of the opcode to which they refer. They may also be entered on a separate line.

The segment override mnemonics are CS:, DS:, ES:, and SS:. The mnemonic for the far return is RETF. String manipulation mnemonics must explicitly state the string size. For example, use MOVSW to move word strings and MOVSB to move byte strings.

The assembler will automatically assemble short, near or far jumps and calls, depending on byte displacement to the destination address. These may be overridden with the NEAR or FAR prefix. For example:

```
0100:0500 JMP  502          ; a 2-byte short jump
0100:0502 JMP  NEAR 505     ; a 3-byte near jump
0100:505  JMP  FAR  50A     ; a 5-byte far jump
```

The NEAR prefix may be abbreviated to NE, but the FAR prefix cannot be abbreviated.

DEBUG cannot tell whether some operands refer to a word memory location or to a byte memory location. In this case, the data type must be explicitly stated with the prefix "WORD PTR" or "BYTE PTR". Acceptable abbreviations are "WO" and "BY". For example:

```
NEG     BYTE PTR [128]
DEC     WO [SI]
```

DEBUG also cannot tell whether an operand refers to a memory location or to an immediate operand. DEBUG uses the common convention that operands enclosed in square brackets refer to memory. For example:

```
MOV    AX,21        ; Load AX with 21H
MOV    AX,[21]      ; Load AX with the
                    ; contents
                    ; of memory location 21H
```

Two popular pseudo-instructions are available with Assemble. The DB opcode will assemble byte values directly into memory. The DW opcode will assemble word values directly into memory. For example:

```
DB       1,2,3,4,"THIS IS AN EXAMPLE"
DB       'THIS IS A QUOTE: "'
DB       "THIS IS A QUOTE: '"

DW       1000,2000,3000,"BACH"
```

Assemble supports all forms of register indirect commands. For example:

```
ADD      BX,34[BP+2].[SI-1]
POP      [BP+DI]
PUSH     [SI]
```

All opcode synonyms are also supported. For example:

```
LOOPZ    100
LOOPE    100

JA       200
JNBE     200
```

For 8087 opcodes, the WAIT or FWAIT must be explicitly specified. For example:

```
FWAIT FADD ST,ST(3)   ; This line assembles
                      ; an FWAIT prefix
LD TBYTE PTR [BX]     ; This line does not
```

NAME

Compare

PURPOSE

Compares the portion of memory specified by <range> to a portion of the same size beginning at <address>.

SYNTAX

```
C<range> <address>
```

COMMENTS

If the two areas of memory are identical, there is no display and DEBUG returns with the MS-DOS prompt. If there are differences, they are displayed in this format:

```
<address1> <byte1> <byte2> <address2>
```

EXAMPLE

The following commands have the same effect:

```
C100,1FF 300

     or

C100L100 300
```

Each command compares the block of memory from 100 to 1FFH with the block of memory from 300 to 3FFH.

NAME

Dump

PURPOSE

Displays the contents of the specified region of memory.

SYNTAX

```
D[<range>]
```

COMMENTS

If a range of addresses is specified, the contents of the range are displayed. If the D command is typed without parameters, 128 bytes are displayed at the first address (DS:100) after the address displayed by the previous Dump command.

The dump is displayed in two portions: a hexadecimal dump (each byte is shown in hexadecimal value) and an ASCII dump (the bytes are shown in ASCII characters). Nonprinting characters are denoted by a period (.) in the ASCII portion of the display. Each display line shows 16 bytes with a hyphen between the eighth and ninth bytes. At times, displays are split in this manual to fit them on the page. Each displayed line begins on a 16-byte boundary.

If you type the command:

```
dcs:100 110
```

DEBUG displays the dump in the following format:

```
04BA:0100 42 45 52 54 41 ... 4E 44 TOM SAWYER
```

If you type the following command:

```
D
```

the display is formatted as described above. Each line of the display begins with an address, incremented by 16 from the address on the previous line. Each subsequent D (typed without parameters) displays the bytes immediately following those last displayed.

If you type the command:

```
DCS:100 L 20
```

the display is formatted as described above, but 20H bytes are displayed.

If then you type the command:

```
DCS:100 115
```

the display is formatted as described above, but all the bytes in the range of lines from 100H to 115H in the CS segment are displayed.

NAME

Enter

PURPOSE

Enters byte values into memory at the specified <address>.

SYNTAX

```
E<address>[ <list>]
```

COMMENTS

If the optional <list> of values is typed, the replacement of byte values occurs automatically. (If an error occurs, no byte values are changed.)

If the <address> is typed without the optional <list>, DEBUG displays the address and its contents, then repeats the address on the next line and waits for your input. At this point, the Enter command waits for you to perform one of the following actions:

1. Replace a byte value with a value you type. Simply type the value after the current value. If the value typed in is not a legal hexadecimal value or if more than two digits are typed, the illegal or extra character is not echoed.

2. Press the Spacebar to advance to the next byte. To change the value, simply type the new value as described in (1.) above. If you space beyond an 8-byte boundary, DEBUG starts a new display line with the address displayed at the beginning.

3. Type a hyphen (-) to return to the preceding byte. If you decide to change a byte behind the current position, typing the hyphen returns the current position to the previous byte. When the hyphen is typed, a new line is started with the address and its byte value displayed.

4. Press the Return key to terminate the Enter command. The Return key may be pressed at any byte position.

EXAMPLE

Assume that the following command is typed:

```
ECS:100
```

DEBUG displays:

```
04BA:0100  EB._
```

To change this value to 41, type 41 as shown:

```
04BA:0100  EB.41_
```

To step through the subsequent bytes, press the Spacebar to see:

```
04BA:0100  EB.41   10.   00.   BC._
```

To change BC to 42:

```
04BA:0100  EB.41   10.   00.   BC.42_
```

Now, realizing that 10 should be 6F, type the hyphen as many times as needed to return to byte 0101 (value 10), then replace 10 with 6F:

```
04BA:0100  EB.41   10.   00.   BC.42-
04BA:0102  00.-
04BA:0101  10.6F_
```

Pressing the Return key ends the Enter command and returns to the DEBUG command level.

NAME

Fill

PURPOSE

Fills the addresses in the <range> with the values in the <list>.

SYNTAX

```
F<range> <list>
```

COMMENTS

If the <range> contains more bytes than the number of values in the <list>, the <list> will be used repeatedly until all bytes in the <range> are filled. If the <list> contains more values than the number of bytes in the <range>, the extra values in the <list> will be ignored. If any of the memory in the <range> is not valid (bad or nonexistent), the error will occur in all succeeding locations.

EXAMPLE

Assume that the following command is typed:

```
F04BA:100 L 100 42 45 52 54 41
```

DEBUG fills memory locations 04BA:100 through 04BA:1FF with the bytes specified. The five values are repeated until all 100H bytes are filled.

NAME

Go

PURPOSE

Executes the program currently in memory.

SYNTAX

```
G[=<address>[ <address>...]]
```

COMMENTS

If only the Go command is typed, the program executes as if the program had run outside DEBUG.

If =<address> is set, execution begins at the address specified. The equal sign (=) is required, so that DEBUG can distinguish the start =<address> from the breakpoint <address>es.

With the other optional addresses set, execution stops at the first <address> encountered, regardless of that address' position in the list of addresses to halt execution or program branching. When program execution reaches a breakpoint, the registers, flags, and decoded instruction are displayed for the last instruction executed. (The result is the same as if you had typed the Register command for the breakpoint address.)

Up to ten breakpoints may be set. Breakpoints may be set only at addresses containing the first byte of an 8086 opcode. If more than ten breakpoints are set, DEBUG returns the BP Error message.

The user stack pointer must be valid and have 6 bytes available for this command. The G command uses an IRET instruction to cause a jump to the program under test. The user stack pointer is set, and the user flags, Code Segment register, and Instruction Pointer are pushed on the user stack. (Thus, if the user stack is not valid or is too small, the operating system may crash.) An interrupt code (0CCH) is placed at the specified breakpoint address(es).

When an instruction with the breakpoint code is encountered, all breakpoint addresses are restored to their original instructions. If execution is not halted at one of the breakpoints, the interrupt codes are not replaced with the original instructions.

EXAMPLE

Assume that the following command is typed:

```
GCS:7550
```

The program currently in memory executes up to the address 7550 in the CS segment. DEBUG then displays registers and flags, after which the Go command is terminated.

After a breakpoint has been encountered, if you type the Go command again, then the program executes just as if you had typed the filename at the MS-DOS command level. The only difference is that program execution begins at the instruction after the breakpoint rather than at the usual start address.

NAME

Hex

PURPOSE

Performs hexadecimal arithmetic on the two parameters specified.

SYNTAX

```
H<value> <value>
```

COMMENTS

First, DEBUG adds the two parameters, then subtracts the second parameter from the first. The results of the arithmetic are displayed on one line; first the sum, then the difference.

EXAMPLE

Assume that the following command is typed:

```
H19F 10A
```

DEBUG performs the calculations and then displays the result:

```
02A9   0095
```

NAME

Input

PURPOSE

Inputs and displays one byte from the port specified by <value>.

SYNTAX

```
I<value>
```

COMMENTS

A 16-bit port address is allowed.

EXAMPLE

Assume that you type the following command:

```
I2F8
```

Assume also that the byte at the port is 42H. DEBUG inputs the byte and displays the value:

```
42
```

NAME

Load

PURPOSE

Loads a file into memory.

SYNTAX

```
L[<address> [<drive:> <record> <record>]]
```

COMMENTS

Set BX:CX to the number of bytes read. The file must have been named either when DEBUG was started or with the N command. Both the DEBUG invocation and the N command format a filename properly in the normal format of a file control block at CS:5C.

If the L command is typed without any parameters, DEBUG loads the file into memory beginning at address CS:100 and sets BX:CX to the number of bytes loaded. If the L command is typed with an address parameter, loading begins at the memory <address> specified. If L is typed with all parameters, absolute disk sectors are loaded, not a file. The <record>s are taken from the <drive:> specified (the drive designation is numeric here--0=A:, 1=B:, 2=C:, etc.); DEBUG begins loading with the first <record> specified, and continues until the number of sectors specified in the second <record> have been loaded.

EXAMPLE

Assume that the following commands are typed:

```
A>DEBUG
-NFILE.COM
```

Now, to load FILE.COM, type:

```
L
```

DEBUG loads the file and then displays the DEBUG prompt. Assume that you want to load only portions of a file or certain records from a disk. To do this, type:

```
L04BA:100 2 0F 6D
```

DEBUG then loads 109 (6D hex) records beginning with logical record number 15 into memory beginning at address 04BA:0100. When the records have been loaded, DEBUG simply returns the - prompt.

If the file has a .EXE extension, it is relocated to the load address specified in the header of the .EXE file: the <address> parameter is always ignored for .EXE files. The header itself is stripped off the .EXE file before it is loaded into memory. Thus the size of an .EXE file on disk will differ from its size in memory.

If the file named by the Name command or specified when DEBUG is started is a .HEX file, then typing the L command with no parameters causes DEBUG to load the file beginning at the address specified in the .HEX file. If the L command includes the option <address>, DEBUG adds the <address> specified in the L command to the address found in the .HEX file to determine the start address for loading the file.

NAME

Move

PURPOSE

Moves the block of memory specified by <range> to the location beginning at the <address> specified.

SYNTAX

```
M<range> <address>
```

COMMENTS

Overlapping moves (i.e., moves where part of the block overlaps some of the current addresses) are always performed without loss of data. Addresses that could be overwritten are moved first. The sequence for moves from higher addresses to lower addresses is to move the data beginning at the block's lowest address and then to work towards the highest. The sequence for moves from lower addresses to higher addresses is to move the data beginning at the block's highest address and to work towards the lowest.

Note that if the addresses in the block being moved will not have new data written to them, the data there before the move will remain. The M command copies the data from one area into another, in the sequence described, and writes over the new addresses. This is why the sequence of the move is important.

EXAMPLE

Assume that you type:

```
MCS:100 110 CS:500
```

DEBUG first moves address CS:110 to address CS:510, then CS:10F to CS:50F, and so on until CS:100 is moved to CS:500. You should type the D command, using the <address> typed for the M command, to review the results of the move.

NAME

Name

PURPOSE

Sets filenames.

SYNTAX

```
N<filename>[<filename>...]
```

COMMENTS

The Name command performs two functions. First, Name is used to assign a filename for a later Load or Write command. Thus, if you start DEBUG without naming any file to be debugged, then the N<filename> command must be typed before a file can be loaded. Second, Name is used to assign filename parameters to the file being debugged. In this case, Name accepts a list of parameters that are used by the file being debugged.

These two functions overlap. Consider the following set of DEBUG commands:

```
-NFILE1.EXE
-L
-G
```

Because of the effects of the Name command, Name will perform the following steps:

1. (N)ame assigns the filename FILE1.EXE to the filename to be used in any later Load or Write commands.

2. (N)ame also assigns the filename FILE1.EXE to the first filename parameter used by any program that is later debugged.

3. (L)oad loads FILE1.EXE into memory.

4. (G)o causes FILE1.EXE to be executed with FILE1.EXE as the single filename parameter (that is, FILE1.EXE is executed as if FILE1.EXE had been typed at the command level).

A more useful chain of commands might look like this:

```
-NFILE1.EXE
-L
-NFILE2.DAT FILE3.DAT
-G
```

Here, Name sets FILE1.EXE as the filename for the subsequent Load command. The Load command loads FILE1.EXE into memory, and then the Name command is used again, this time to specify the parameters to be used by FILE1.EXE. Finally, when the Go command is executed, FILE1.EXE is executed as if FILE1 FILE2.DAT FILE3.DAT had been typed at the MS-DOS command level. Note that if a Write command were executed at this point, then FILE1.EXE--the file being debugged--would be saved with the name FILE2.DAT! To avoid such undesired results, you should always execute a Name command before either a Load or a Write.

There are four regions of memory that can be affected by the Name command:

CS:5C	FCB for file 1
CS:6C	FCB for file 2
CS:80	Count of characters
CS:81	All characters typed

A File Control Block (FCB) for the first filename parameter given to the Name command is set up at CS:5C. If a second filename parameter is typed, then an FCB is set up for it beginning at CS:6C. The number of characters typed in the Name command (exclusive of the first character, "N") is given at location CS:80. The actual stream of characters given by the Name command (again, exclusive of the letter "N") begins at CS:81. Note that this stream of characters may contain switches and delimiters that would be legal in any command typed at the MS-DOS command level.

EXAMPLE

A typical use of the Name command is:

```
DEBUG PROG.COM
-NPARAM1 PARAM2/C
-G
-
```

In this case, the Go command executes the file in memory as if the following command line had been typed:

```
PROG PARAM1 PARAM2/C
```

Testing and debugging therefore reflect a normal runtime environment for PROG.COM.

NAME

Output

PURPOSE

Sends the <byte> specified to the output port specified by <value>.

SYNTAX

```
O<value> <byte>
```

COMMENTS

A 16-bit port address is allowed.

EXAMPLE

Type:

```
O2F8 4F
```

DEBUG outputs the byte value 4F to output port 2F8.

NAME

Quit

PURPOSE

Terminates the DEBUG utility.

SYNTAX

```
Q
```

COMMENTS

The Q command takes no parameters and exits DEBUG without saving the file currently being operated on. You are returned to the MS-DOS command level.

EXAMPLE

To end the debugging session, type:

```
Q<Return>
```

DEBUG has been terminated, and control returns to the MS-DOS command level.

NAME

Register

PURPOSE

Displays the contents of one or more CPU registers.

SYNTAX

R[<register-name>]

COMMENTS

If no <register-name> is typed, the R command dumps the register save area and displays the contents of all registers and flags.

If a register name is typed, the 16-bit value of that register is displayed in hexadecimal, and then a colon appears as a prompt. You then either type a <value> to change the register, or simply press the Return key if no change is wanted.

The only valid <register-name>s are:

AX	BP	SS	
BX	SI	CS	
CX	DI	IP	(IP and PC both refer
DX	DS	PC	to the Instruction
SP	ES	F	Pointer.)

Any other entry for <register-name> results in a BR Error message.

If F is entered as the <register-name>, DEBUG displays each flag with a two-character alphabetic code. To alter any flag, type the opposite two-letter code. The flags are either set or cleared.

The flags are listed below with their codes for SET and CLEAR:

FLAG NAME	SET	CLEAR
Overflow	OV	NV
Direction	DN Decrement	UP Increment
Interrupt	EI Enabled	DI Disabled
Sign	NG Negative	PL Plus
Zero	ZR	NZ
Auxiliary Carry	AC	NA
Parity	PE Even	PO Odd
Carry	CY	NC

Whenever you type the command RF, the flags are displayed in the order shown above in a row at the beginning of a line. At the end of the list of flags, DEBUG displays a hyphen (-). You may enter new flag values as alphabetic pairs. The new flag values can be entered in any order. You do not have to leave spaces between the flag entries. To exit the R command, press the Return key. Flags for which new values were not entered remain unchanged.

If more than one value is entered for a flag, DEBUG returns a DF Error message. If you enter a flag code other than those shown above, DEBUG returns a BF Error message. In both cases, the flags up to the error in the list are changed; flags at and after the error are not.

At startup, the segment registers are set to the bottom of free memory, the Instruction Pointer is set to 0100H, all flags are cleared, and the remaining registers are set to zero.

EXAMPLE

Type:

```
R
```

DEBUG displays all registers, flags, and the decoded instruction for the current location. If the location is CS:11A, then the display will look similar to this:

```
AX=0E00 BX=00FF CX=0007 DX=01FF SP=039D BP=0000
SI=005C DI=0000 DS=04BA ES=04BA SS=04BA CS=04BA
IP=011A   NV UP DI NG NZ AC PE NC
04BA:011A  CD21          INT      21
```

If you type:

```
RF
```

DEBUG will display the flags:

```
NV UP DI NG NZ AC PE NC - _
```

Now, type any valid flag designation, in any order, with or without spaces.

For example:

```
NV UP DI NG NZ AC PE NC - PLEICY<RETURN>
```

DEBUG responds only with the DEBUG prompt. To see the changes, type either the R or RF command:

```
RF
NV UP EI PL NZ AC PE CY - _
```

Press the Return key to leave the flags this way, or to specify different flag values.

NAME

Search

PURPOSE

Searches the <range> specified for the <list> of bytes specified.

SYNTAX

```
S<range> <list>
```

COMMENTS

The <list> may contain one or more bytes, each separated by a space or comma. If the <list> contains more than one byte, only the first address of the byte string is returned. If the <list> contains only one byte, all addresses of the byte in the <range> are displayed.

EXAMPLE

If you type:

```
SCS:100 110 41
```

DEBUG displays a response similar to this:

```
04BA:0104
04BA:010D
-type:
```

NAME

Trace

PURPOSE

Executes one instruction and displays the contents of all registers and flags, and the decoded instruction.

SYNTAX

```
T[=<address>][ <value>]
```

COMMENTS

If the optional =<address> is typed, tracing occurs at the =<address> specified. The optional <value> causes DEBUG to execute and trace the number of steps specified by <value>.

The T command uses the hardware trace mode of the 8086 or 8088 microprocessor. Consequently, you may also trace instructions stored in ROM (Read Only Memory).

EXAMPLE

Type:

```
T
```

DEBUG returns a display of the registers, flags, and decoded instruction for that one instruction. Assume that the current position is 04BA:011A; DEBUG might return the display:

```
AX=0E00 BX=00FF CX=0007 DX=01FF SP=039D BP=0000
SI=005C DI=0000 DS=04BA ES=04BA SS=04BA CS=04BA
IP=011A  NV UP DI NG NZ AC PE NC
04BA:011A  CD21          INT     21
```

If you type:

```
T=011A 10
```

DEBUG executes sixteen (10 hex) instructions beginning at 011A in the current segment, and then displays all registers and flags for each instruction as it is executed. The display scrolls away until the last instruction is executed. Then the display stops, and you can see the register and flag values for the last few instructions performed. Remember that Control-S suspends the display at any point, so that you can study the registers and flags for any instruction.

NAME

Unassemble

PURPOSE

Disassembles bytes and displays the source statements that correspond to them, with addresses and byte values.

SYNTAX

U[<range>]

COMMENTS

The display of disassembled code looks like a listing for an assembled file. If you type the U command without parameters, 20 hexadecimal bytes are disassembled at the first address after that displayed by the previous Unassemble command. If you type the U command with the <range> parameter, then DEBUG disassembles all bytes in the range. If the <range> is given as an <address> only, then 20H bytes are disassembled.

EXAMPLE

Type:

```
U04BA:100 L10
```

DEBUG disassembles 16 bytes beginning at address 04BA:0100:

```
04BA:0100  206472     AND  [SI+72],AH
04BA:0103  69         DB   69
04BA:0104  7665       JBE  016B
04BA:0106  207370     AND  [BP+DI+70],DH
04BA:0109  65         DB   65
04BA:010A  63         DB   63
04BA:010B  69         DB   69
04BA:010C  66         DB   66
04BA:010D  69         DB   69
04BA:010E  63         DB   63
04BA:010F  61         DB   61
```

If you type:

```
U04ba:0100 0108
```

the display shows:

```
04BA:0100  206472    AND   [SI+72],AH
04BA:0103  69        DB    69
04BA:0104  7665      JBE   016B
04BA:0106  207370    AND   [BP+DI+70],DH
```

If the bytes in some addresses are altered, the disassembler alters the instruction statements. The U command can be typed for the changed locations, the new instructions viewed, and the disassembled code used to edit the source file.

NAME

Write

PURPOSE

Writes the file being debugged to a disk file.

SYNTAX

```
W[<address>[ <drive:> <record> <record>]]
```

COMMENTS

If you type W with no parameters, BX:CX must already be set to the number of bytes to be written; the file is written beginning from CS:100. If the W command is typed with just an address, then the file is written beginning at that address. If a G or T command has been used, BX:CX must be reset before using the Write command without parameters. Note that if a file is loaded and modified, the name, length, and starting address are all set correctly to save the modified file (as long as the length has not changed).

The file must have been named either with the DEBUG invocation command or with the N command (refer to the Name command earlier in this manual). Both the DEBUG invocation and the N command format a filename properly in the normal format of a file control block at CS:5C.

If the W command is typed with parameters, the write begins from the memory address specified; the file is written to the <drive:> specified (the drive designation is numeric here--0=A:, 1=B:, 2=C:, etc.); DEBUG writes the file beginning at the logical record number specified by the first <record>; DEBUG continues to write the file until the number of sectors specified in the second <record> have been written.

Warning

Writing to absolute sectors is EXTREMELY dangerous because the process bypasses the file handler.

EXAMPLE

Type:

```
W
```

DEBUG will write the file to disk and then display the DEBUG prompt. Two examples are shown below.

```
W
-_
```

```
WCS:100 1 37 2B
```

DEBUG writes out the contents of memory, beginning with the address CS:100 to the disk in drive B:. The data written out starts in disk logical record number 37H and consists of 2BH records. When the write is complete, DEBUG displays the prompt:

```
WCS:100 1 37 2B
-_
```

8.5 ERROR MESSAGES

During the DEBUG session, you may receive any of the following error messages. Each error terminates the DEBUG command under which it occurred, but does not terminate DEBUG itself.

ERROR CODE	DEFINITION
BF	Bad flag You attempted to alter a flag, but the characters typed were not one of the acceptable pairs of flag values. See the Register command for the list of acceptable flag entries.
BP	Too many breakpoints You specified more than ten breakpoints as parameters to the G command. Retype the Go command with ten or fewer breakpoints.
BR	Bad register You typed the R command with an invalid register name. See the Register command for the list of valid register names.
DF	Double flag You typed two values for one flag. You may specify a flag value only once per RF command.

Appendix A
Instructions for Users with Single-drive Systems

APPENDIX A

INSTRUCTIONS FOR USERS WITH SINGLE-DRIVE SYSTEMS

Note

Information in this appendix is installation-dependent and may not be implemented on every manufacturer's machine.

If you have only one disk drive, you type MS-DOS commands just as if you had two or more drives on your system.

You should think of your one-drive system as having two drives (drive A and drive B). But instead of A and B representing two physical drives, the A and B represent disks.

If you specify drive B when the "drive A disk" was last used, MS-DOS prompts you to insert the disk for drive B. For example:

```
A> copy command.com b:
Insert diskette for drive B:
and strike any key when ready
  1 File(s) copied
A>_
```

If you specify drive A when the "drive B disk" was last used, MS-DOS prompts you to change disks again. This time, MS-DOS prompts you to insert the "drive A disk."

Use the same procedure when using a batch file to execute commands. MS-DOS waits for you to insert the appropriate disk and press any key before it continues. You will be prompted to insert a disk.

Important

The letter displayed in the system prompt represents the default drive where MS-DOS looks to find a file whose name is entered without a drive name. The letter in the system prompt does not represent the last disk used.

For example, assume that A is the default drive. If the last command performed was:

```
    dir b:
```

MS-DOS believes the "drive B disk" is still in the drive. However, the system prompt is still A>, because A is still the default drive. If you type:

```
    dir
```

MS-DOS prompts you for the "drive A disk" because drive A is the default drive, and you did not specify another drive in the Dir command.

Appendix B
Disk and Device Errors

APPENDIX B

DISK AND DEVICE ERRORS

If a disk or device error occurs at any time during a command or program, MS-DOS displays an error message in the following format:

```
<type> <action> drive <x>
Abort,Ignore,Retry:_
```

In this message,<type> may be one of the following:

```
Write protect error
Bad unit error
Not ready error
Bad command error
Data error
Bad call format error
Seek error
Non-DOS disk error
Sector not found error
No paper error
Write fault error
Read fault error
General failure
Sharing violation
Lock violation
Invalid disk change
FCB unavailable
```

The <action> may be either of the following:

```
READING
WRITING
```

The drive indicates the drive in which the error occurred.

MS-DOS waits for you to respond in one of the following ways:

A <u>Abort</u>. End the program requesting the disk read or write.

I <u>Ignore</u>. Ignore the bad sector and pretend the error did not occur.

R <u>Retry</u>. Repeat the operation. This response should be used if you have corrected the error (such as with Not ready or Write protect errors).

Usually, you will want to recover by typing responses in this order:

```
R   (to try again)
A   (to terminate program and try a new disk)
```

One other error message might be related to faulty disk read or write:

```
FILE ALLOCATION TABLE BAD FOR DRIVE x
```

This message means that the copy in memory of one of the allocation tables has pointers to nonexistent blocks. Possibly the disk was incorrectly formatted or not formatted before use. If this error persists, the disk is currently unusable and must be formatted prior to use.

Appendix C
ANSI Escape Sequences

APPENDIX C

ANSI ESCAPE SEQUENCES

Note

Information in this appendix is installation-dependent and may not be implemented on every manufacturer's machine.

An ANSI escape sequence is a series of characters (beginning with an escape character or keystroke) that you can use to define functions to MS-DOS. Specifically, you can reassign keys, change graphics functions, and affect cursor movement.

This appendix explains how the ANSI escape sequences are defined for MS-DOS. Examples of how to use ANSI escape sequences are included at the end of this appendix.

NOTES:

1. MS-DOS uses a default value when you do not specify a value or when you specify zero.
2. Pn represents "numeric parameter." This is a decimal number specified with ASCII digits.
3. Ps represents "selective parameter." This is any decimal number that is used to select a subfunction. Multiple subfunctions may be selected by separating the parameters with semicolons.
4. Pl represents "line parameter." This is a decimal number specified with ASCII digits.
5. Pc represents "column parameter." This is a decimal number specified with ASCII digits.

C.1 CURSOR FUNCTIONS

The following escape sequences affect the cursor position on the screen.

CUP - Cursor Position

```
ESC [ Pl ; Pc H
```

HVP - Horizontal & Vertical Position

```
ESC [ Pl ; Pc f
```

CUP and HVP move the cursor to the position specified by the parameters. The first parameter specifies the line number, and the second parameter specifies the column number. The default value is 1. When no parameters are specified, the cursor moves to the home position.

CUU - Cursor Up

```
ESC [ Pn A
```

This sequence moves the cursor up one line without changing columns. The value of Pn sets the number of lines moved. The default value for Pn is 1. MS-DOS ignores the CUU sequence if the cursor is already on the top line.

CUD - Cursor Down

```
ESC [ Pn B
```

This sequence moves the cursor down one line without changing columns. The value of Pn sets the number of lines moved. The default value for Pn is 1. MS-DOS ignores the CUD sequence if the cursor is already on the bottom line.

CUF - Cursor Forward

```
ESC [ Pn C
```

The CUF sequence moves the cursor forward one column without changing lines. The value of Pn sets the number of columns moved. The default value for Pn is 1. MS-DOS ignores the CUF sequence if the cursor is already in the far right column.

CUB - Cursor Backward

```
ESC [ Pn D
```

This escape sequence moves the cursor back one column without changing lines. The value of Pn sets the number of columns moved. The default value for Pn is 1. MS-DOS ignores the CUB sequence if the cursor is already in the far left column.

DSR - Device Status Report

```
ESC [ 6 n
```

The console driver will output a CPR sequence (see below) on receipt of the DSR escape sequence.

CPR - Cursor Position Report (from console driver to system)

```
ESC [ Pn ; Pn R
```

The CPR sequence reports current cursor position using standard input. The first parameter specifies the current line and the second parameter specifies the current column.

SCP - Save Cursor Position

```
ESC [ s
```

The current cursor position is saved. This cursor position can be restored with the RCP sequence (see below).

RCP - Restore Cursor Position

```
ESC [ u
```

This sequence restores the cursor position to the value it had when the console driver received the SCP sequence.

C.2 ERASING

The following escape sequences affect erase functions.

ED - Erase Display

```
    ESC [ 2 J
```

The ED sequence erases the screen, and the cursor goes to the home position.

EL - Erase Line

```
    ESC [ K
```

This sequence erases from the cursor to the end of the line (including the cursor position).

C.3 MODES OF OPERATION

The following escape sequences affect screen graphics.

SGR - Set Graphics Rendition

```
    ESC [ Ps ; ... ; Ps m
```

The SGR escape sequence calls the graphic functions specified by the parameters described below. The graphic functions remain until the next occurrence of an SGR escape sequence.

Parameter	Parameter Function	
0	All Attributes off	
1	Bold on	
2	Faint on	
3	Italic on	
5	Blink on	
6	Rapid blink on	
7	Reverse Video on	
8	Concealed on	(ISO 6429 standard)
30	Black foreground	(ISO 6429 standard)
31	Red foreground	(ISO 6429 standard)
32	Green foreground	(ISO 6429 standard)
33	Yellow foreground	(ISO 6429 standard)
34	Blue foreground	(ISO 6429 standard)
35	Magenta foreground	(ISO 6429 standard)
36	Cyan foreground	(ISO 6429 standard)

37	White foregound	(ISO 6429 standard)
40	Black background	(ISO 6429 standard)
41	Red background	(ISO 6429 standard)
42	Green background	(ISO 6429 standard)
43	Yellow background	(ISO 6429 standard)
44	Blue background	(ISO 6429 standard)
45	Magenta background	(ISO 6429 standard)
46	Cyan background	(ISO 6429 standard)
47	White background	(ISO 6429 standard)
48	Subscript	
49	Superscript	

SM - Set Mode

```
     ESC [ = Ps h
or   ESC [ = h
or   ESC [ = 0 h
or   ESC [ ? 7 h
```

The SM escape sequence changes the screen width or type to one of the following:

Parameter	Parameter Function
0	40 x 25 black and white
1	40 x 25 color
2	80 x 25 black and white
3	80 x 25 color
4	320 x 200 color
5	320 x 200 black and white
6	640 x 200 black and white
7	wrap at end of line

RM - Reset Mode

```
     ESC [ = Ps l
or   ESC [ = l
or   ESC [ = 0 l
or   ESC [ ? 7 l
```

Parameters for RM are the same as for SM (Set Mode), except that parameter 7 resets the wrap at the end of line mode.

C.4 KEYBOARD REASSIGNMENT

Although not part of the ANSI 3.64-1979 or ISO 6429 standard, the following keyboard reassignments are compatible with these standards.

The control sequence is:

```
     ESC [ Pn ; Pn ; ... Pn p
or   ESC [ "string" ; p
or   ESC [ Pn ; "string" ; Pn ; Pn ; "string" ; Pn p
or   any other combination of strings and decimal numbers
```

The final code in the control sequence (p) is one reserved for private use by the ANSI 3.64-1979 standard.

The first ASCII code in the control sequence defines which code is being mapped. The remaining numbers define the sequence of ASCII codes generated when this key is intercepted. Note that there is one exception: if the first code in the sequence is zero (NUL), then the first and second code make up an extended ASCII redefinition.

Examples:

1. Reassign the Q and q key to the A and a key (and vice versa):

```
ESC [ 6 5 ; 8 1 p                  A becomes Q
ESC [ 9 7 ; 1 1 3 p                a becomes q
ESC [ 8 1 ; 6 5 p                  Q becomes A
ESC [ 1 1 3 ; 9 7 p                q becomes a
```

2. Reassign the F10 key to a Dir command followed by a carriage return:

```
ESC [ 0 ; 6 8 ; " d i r " ; 1 3 p
```

The 0;68 is the extended ASCII code for the F10 key; 13 decimal is a carriage return.

Appendix D
How to Configure Your System

APPENDIX D

HOW TO CONFIGURE YOUR SYSTEM

D.1 WHAT IS A CONFIGURATION FILE?

In many cases, there are installation-specific settings for MS-DOS that need to be configured at system startup. An example of this is a standard device driver, such as an online printer.

The configuration file, named CONFIG.SYS, is simply a file that has certain commands for MS-DOS at startup. Each time you start MS-DOS, MS-DOS searches the root directory of the drive it was started from for a file named CONFIG.SYS. The CONFIG.SYS file allows you to configure your system with a minimum of effort. For example, you can add device drivers to your system at startup by a simple command in the CONFIG.SYS file.

D.2 CHANGING THE CONFIG.SYS FILE

If there is not a CONFIG.SYS file on the MS-DOS disk, you can use the MS-DOS editor, EDLIN, to create a file; then save it on the MS-DOS disk in your <u>root</u> directory.

D.3 CONFIG.SYS COMMANDS

The following commands are used in the CONFIG.SYS file:

Break	Sets Control-C check.
Buffers	Sets the number of sector buffers.
Country	Allows for international time, date and currency.
Device	Installs the device driver into the system.

Drivparm	Defines parameters for block devices.
FCBS	Specifies the number of FCBs that can be concurrently open.
Files	Sets the number of open files that can access certain MS-DOS system calls.
Lastdrive	Sets the maximum number of drives that you may access.
Shell	Begins execution of the shell from a specific file.

These commands are described in detail on the following pages. For a sample CONFIG.SYS file, see Section D.4 at the end of this appendix.

NAME

Break

PURPOSE

Sets Control-C check.

SYNTAX

```
break=[ON]

  or

break=[OFF]
```

COMMENTS

Depending on the program you are running, Control-C may be used to stop an activity (for example, to stop sorting a file). Normally, MS-DOS checks to see if Control-C has been typed while it is reading from the keyboard, writing to the screen, or to a printer. Setting Break to "on" allows Control-C checking to be extended to other functions such as disk reads and writes.

EXAMPLE

```
break=off
```

NAME

Buffers

PURPOSE

Allows you to set the number of disk buffers that MS-DOS allocates in memory at the time you start the system.

SYNTAX

```
buffers=<x>
```

COMMENTS

A disk buffer is a block of memory where MS-DOS can hold data being read from or written to a disk when the amount of data is not an exact multiple of sector size.

The default number of buffers is 2. For applications such as word processors, a number between 10 and 20 will provide the best performance. If you plan to create a lot of subdirectories, increase the buffers value to between 20 and 30.

EXAMPLE

```
buffers=10
```

NAME

Country

PURPOSE

This number allows MS-DOS to use international time, date, currency, and case conversion.

SYNTAX

```
country=<x>
```

COMMENTS

The default and allowed country values are set by the equipment manufacturer. The following table shows the possible values for <x>:

Value	Country
001	United States
031	Netherlands
032	Belgium
033	France
034	Spain
039	Italy
041	Switzerland
044	United Kingdom
045	Denmark
046	Sweden
047	Norway
049	Germany
061	Australia
358	Finland
972	Israel

EXAMPLE

```
country=033
```

This example sets international currency, time, date, and case conversion to French (France).

NAME

Device

PURPOSE

This command installs the device driver in the specified pathname to the system list.

SYNTAX

```
device=[<drive:>]<pathname>
```

COMMENTS

If you plan to use the ANSI escape sequences described in Appendix C, you should create a CONFIG.SYS file with the following command:

```
device=ansi.sys
```

This command causes MS-DOS to replace all keyboard input and screen output support with the ANSI escape sequences.

The standard installable device drivers provided with MS-DOS are ANSI.SYS, DRIVER.SYS, and RAMDRIVE.SYS. For more information on ANSI.SYS, refer to Appendix C. For more information on DRIVER.SYS and RAMDRIVE.SYS, refer to Appendix G, "Installable Device Drivers."

NAME

Drivparm

PURPOSE

This command allows you to define parameters for block devices when you start MS-DOS, overriding the original MS-DOS device driver settings.

SYNTAX

```
drivparm=/d:<dd>  [/c]  [/f:<ff>]  [/h:<hh>]  [/n]
[/s:<sss>] [/t<ttt>]
```

COMMENTS

Setting Drivparm overrides any previous block device driver definitions.

The <dd> parameter on the /d switch specifies a logical drive number between 0 and 255. This means that drive number 0=A, 1=B, 2=C, etc.

The /c switch specifies that changeline (doorlock) support is required.

The <ff> option on the /f switch specifies the form factor index where:

```
0 = 320/360 KB
1 = 1.2 MB
2 = 720 KB
3 = 8" single density
4 = 8" double density
5 = Hard disk
6 = Tape drive
7 = Other
```

If you do not specify the /f switch, Drivparm uses a default of 720KB.

The <hh> option on the /h switch specifies the maximum head number. Its value can range from 1 to 99.

The /n switch specifies a non-removable block device.

The <ss> option on the /s switch specifies the number of sectors per track. Its value can range from 1 to 99.

The <ttt> option on the /t switch specifies the number of tracks per side on the block device. Its value can range from 1 to 999.

EXAMPLE

You might have a computer with an internal tape drive unit on drive D: that is configured at boot time to write 20 tracks of 40 sectors per track. If you want to reconfigure this tape drive to write 10 tracks of 99 sectors each, you can put the following line in your CONFIG.SYS file:

```
drivparm=/d:3 /f:6 /h:1 /s:99 /t:10
```

This overrides the default device driver settings, and supports a tape drive as drive D (in this case the logical and physical drive numbers are identical). This tape drive has 1 head, and supports a tape format of 10 tracks and 99 sectors per track. (This assumes that the device driver for the tape device supports this configuration of tracks and sectors.) You might want to use this method to create a tape that you can read on another computer that can only read this alternate format.

NAME

FCBS

PURPOSE

This command allows you to determine the number of FCBs (File Control Blocks) that can be concurrently open.

SYNTAX

```
fcbs=<x>,<y>
```

COMMENTS

<x> represents the number of files opened by File Control Blocks that can be open at any one time. The default value for <x> is 4. Allowed values are from 1 to 255.

<y> specifies the number of files opened by FCBs that MS-DOS cannot close automatically if an application tries to open more than <x> files by FCBs. The first <y> files opened by FCBs are protected from being closed. The default value is 0. Allowed values are from 1 to 255.

EXAMPLE

```
fcb=4,2
```

NAME

Files

PURPOSE

This command sets the number of open file handles compatible with XENIX(R) that the MS-DOS system calls can access.

SYNTAX

```
files=<x>
```

COMMENTS

<x> represents the number of open file handles that the system calls can access. System calls 2FH through 60H are compatible with the XENIX operating system.

<x> can be any number between 8 and 255. The default value is 8. Any value higher than 20 serves no function.

EXAMPLE

```
files=20
```

NAME

Lastdrive

PURPOSE

This command sets the maximum number of drives that you may access.

SYNTAX

```
lastdrive=<x>
```

COMMENTS

<x> can be any letter from A to Z.

This value represents the last valid drive that MS-DOS will accept. The default value is E. The minumum number is equal to the number of drives you have installed on your computer.

This command is only useful in a network environment. At startup, MS-DOS recognizes five drive letters: A-E, no matter how many physical drives you have on your system. A network redirection must occur to make any of the extra drives defined by Lastdrive valid.

You should note that MS-DOS allocates a data structure for each drive that you specify, so you shouldn't specify more drives than necessary.

EXAMPLE

```
lastdrive=m
```

This sets the last drive to drive "M" unless you have added an external logical device with DRIVER.SYS. See Appendix G, "Installable Device Drivers," for more information on DRIVER.SYS.

NAME

Shell

PURPOSE

This command begins execution of the shell (top-level command processor) from a file defined by the specified pathname.

SYNTAX

```
shell=[<drive:>]<pathname>
```

COMMENTS

This command should be used by system programmers who write their own command processor (the MS-DOS file named COMMAND.COM). MS-DOS will start the processor specified in <pathname> instead of reading the standard COMMAND.COM.

EXAMPLE

```
shell=\bin\command.com /e:3000 a:\bin /p
```

This example uses the COMMAND.COM shell in the \BIN directory with an environment size of 3000 bytes.

D.4 SAMPLE CONFIG.SYS FILE

A typical configuration file might look like this:

```
     buffers=10
     files=10
     device=\bin\network.sys
     break=on
     shell=a:\bin\command.com /e:3000 a:\bin /p
     lastdrive=e
```

Note that the Buffers and Files commands are set to 10. MS-DOS will search for the pathname \BIN\NETWORK.SYS to find the device that is being added to the system. This file is usually supplied on disk with your device. Make sure that you save the device file in the pathname that you specify in the Device command.

This file also sets the MS-DOS command EXEC to the COMMAND.COM file located on disk in drive A in the \BIN directory. The /e switch sets the size of the environment to 3000 bytes. The A:\BIN tells COMMAND.COM where to look for itself when it needs to reread from disk. The /p switch tells COMMAND.COM that it is the first program running on the system so that it can process the MS-DOS Exit command. (Refer to Command in Chapter 3, "MS-DOS Commands," for more information on the command processor.)

The last logical drive on the system is drive E unless you have added an external logical device with DRIVER.SYS. See Appendix G, "Installable Device Drivers," for more information on DRIVER.SYS.

Appendix E
MS-DOS Message Directory

APPENDIX E

MS-DOS MESSAGE DIRECTORY

Abort edit (Y/N)?
[EDLIN]

MS-DOS displays this message when you choose the Quit (Q) command in EDLIN. The Quit command exits the editing session <u>without</u> saving any editing changes. Specify Y (for Yes) or N (for No).

Abort, Retry, Ignore?
[MS-DOS]

If a disk or device error occurs at any time during a command or program, MS-DOS returns this message and asks you to abort the command or program, retry it, or ignore the error.

Access denied
[Replace] [Xcopy]

You tried to replace a write-protected, read-only, or locked file.

All files cancelled by operator
[Print]

MS-DOS displays this message when you specify the /t switch with the Print command.

All specified files are contiguous
[Chkdsk]

All files are allocated contiguously on the disk without fragmentation.

Allocation error in file, size adjusted
[Chkdsk]

The size of the file indicated in the directory was not consistent with the amount of data actually allocated to the file.

Are you sure (Y,N)?
[MS-DOS]

MS-DOS displays this message if you try to delete *.* (all files in the working directory). Specify Y (for Yes) or N (for No).

Attempted write-protect violation
[Format]

The disk you are trying to format is write protected.

Bad call format reading drive (x:)
[MS-DOS]

Device error. See Appendix B.

Bad call format writing drive (x:)
[MS-DOS]

Device error. See Appendix B.

Bad command error reading drive (x:)
[MS-DOS]

Device error. See Appendix B.

Bad command error writing drive (x:)
[MS-DOS]

Device error. See Appendix B.

Bad command or file name
[MS-DOS]

The command cannot find the program you asked it to run. You either mistyped the filename, or the file does not exist on the disk.

Bad or missing (filename)
[MS-DOS]

You specified an invalid device in the CONFIG.SYS file. Check the accuracy of the Device command in the CONFIG.SYS file.

Bad or missing Command Interpreter
[MS-DOS]

MS-DOS cannot find the COMMAND.COM file on the disk; either the file is missing from the root directory, or the file is invalid. Either restart the system or copy the COMMAND.COM file from your backup MS-DOS master disk onto the disk used to start MS-DOS. You will also receive this message if COMMAND.COM has been moved from the directory it was originally in when you started MS-DOS.

Bad Partition Table
[Format]

This message means that there is no DOS partition on the hard disk. You must run Fdisk to create a DOS partition on your hard disk.

Bad unit error reading drive (x:)
[MS-DOS]

Device error. See Appendix B.

BREAK is off (or on)
[MS-DOS]

This message tells you the current setting of Break.

Cannot CHDIR to (filename) -
tree past this point not processed
[Chkdsk]

Chkdsk is traveling the tree structure of the directory and is unable to go to the specified directory. All subdirectories underneath this directory will not be verified.

Cannot CHDIR to root
Processing cannot continue
[Chkdsk]

Chkdsk is traveling the tree structure of the directory and is unable to return to the root directory. Chkdsk is not able to continue checking the remaining subdirectories to the root.

Cannot CHKDSK a Network dri[illegible]
[Chkdsk]

You cannot check drives that are redirected over the network.

Cannot CHKDSK a SUBSTed or ASSIGNed drive
[Chkdsk]
You cannot check drives that have been substituted or assigned.

Cannot DISKCOMP to or from
a network drive
[Diskcomp]
You cannot compare disks on drives that have been redirected over the network.

Cannot DISKCOMP to or from
an ASSIGNed or SUBSTed drive
[Diskcomp]
One of the drives that you specified is a drive that you created using the ASSIGN or SUBST command.

Cannot DISKCOPY to or from
a network drive
[Diskcopy]
You cannot copy disks to or from drives that have been redirected over the network.

Cannot do binary reads from a device
[Copy]
This message appears during Copy command processing. The Copy cannot be done in binary mode when you are copying from a device. Do not use the /b switch, or else specify an ASCII copy with the /a switch.

Cannot edit .BAK file --rename file
[EDLIN]
You attempted to edit a backup copy created by EDLIN. Either rename the file or copy the .BAK file and give it a different extension.

Cannot format an ASSIGNed or SUBSTed drive
[Format]
You attempted to format a drive which is actually mapped to another drive by the Assign or Subst command. Run Assign or Subst again and clear all drive assignments.

Cannot FORMAT a Network drive
[Format]

You cannot format drives that are redirected over the Network.

Cannot label a network drive
[Label]

You cannot label a drive that is shared on a network server station.

Cannot LABEL a SUBSTed or ASSIGNed drive
[Label]

You cannot label a drive if has been substituted with the SUBST command or assigned with the ASSIGN command.

Cannot open (filename)
[Print]

Either MS-DOS cannot find the specified file to print or the file does not exist. Check the command for a valid filename.

Cannot perform a cyclic copy
[Xcopy]

When you are using the /s switch, you may not specify a target that is a subdirectory of the source.

Cannot recover . entry, processing continued
[Chkdsk]

The "." entry (working directory) is defective.

Cannot Recover a Network drive
[Recover]

You cannot recover files on drives that are redirected over the Network.

Cannot recover .. entry
[Chkdsk]

The ".." entry (parent directory) is defective.

Cannot SUBST a network drive
[Subst]

You cannot substitute drives that are redirected over the network.

Cannot SYS to a Network drive
[Sys]
You cannot transfer the system files to drives that are redirected over the network.

Cannot use PRINT - Use NET PRINT
[Print]
You must use the Net Print command to print files.

Cannot XCOPY to a reserved device
[Xcopy]
You cannot copy files to a device.

Cannot XCOPY from a reserved device
[Xcopy]
You cannot copy from a device to a file.

CHDIR .. failed, trying alternate method
[Chkdsk]
In traveling the tree structure, Chkdsk was not able to return to a parent directory. It will try to return to that directory by starting over at the root and traveling down.

Compare another diskette (Y/N)?
[Diskcomp]
Diskcomp displays this message when it has completed its comparison of the disks. Press Y if you want to compare more disks.

Compare error on
side (NNN), track (nnn)
[Diskcomp]
Diskcomp found an error on side NNN, track nnn.

Compare OK
[Diskcomp]
Diskcomp displays this message if the disks are identical.

Compare process ended
[Diskcomp]
Diskcomp displays this message if a fatal error occured during the comparison.

Comparing (tt) tracks
(xx) sectors per track, (y) side(s)
[Diskcomp]

This message confirms the format of the disks that you are comparing.

COM port does not exist
[Mode]

You have specified an invalid COM port.

Content of destination lost before copy
[MS-DOS]

A file to be used as a source file to the Copy command has been overwritten prior to completion of the copy. Example:

```
copy a + b b
```

which destroys B before it can be copied.

Convert lost chains to files (Y/N)?
[Chkdsk]

If you respond Y to this prompt, Chkdsk will recover the lost blocks it found when checking the disk. Chkdsk will create a directory entry and a file for you with the filename FILEnnnn.CHK. If you respond N, Chkdsk frees the lost blocks so they can be reallocated.

Copy complete
Copy another (Y/N)?
[Diskcopy]

Diskcopy has completed processing. Respond Y if you wish to copy another disk. Respond N if you do not wish to copy another disk.

Copy not completed
[Diskcopy]

Diskcopy could not copy the entire disk.

Copying...
[Diskcopy]

This message indicates that Diskcopy is copying a disk.

Copyright 1981,82,83,84,85,86 Microsoft Corp.
[MS-DOS]

This message appears on most MS-DOS utility and command banners.

Corrections will not be written to disk
[Chkdsk]

There are errors on the disk, but you did not specify the /f switch.

Current date is (mm-dd-yy)
[MS-DOS]

This message is displayed in response to the Date command.

Current time is (hh:mm:ss.hh)
[MS-DOS]

This message is displayed in response to the Time command.

Data error reading drive (x:)
[MS-DOS]

Device error. See Appendix B.

Delete current volume label (Y/N)?
[Label]

If a current volume label exists, Label displays this message when you press the Return key in response to the prompt to enter the new volume label for this disk. If you want to delete the volume label, press Y (for Yes) otherwise, press N (for No).

DEVICE Support Not Present
[Diskcomp]

The disk drive does not support MS-DOS 3.2 device control.

Directory is joined
[Chkdsk]

Chkdsk does not process directories which are joined.

Directory not empty
[Join]

You can only join onto a directory which is empty.

Directory is totally empty,
no . or ..
[Chkdsk]

The specified directory does not contain references to working and parent directories. Delete the specified directory and recreate it.

Disk error reading drive (x:)
[MS-DOS]

Device error. See Appendix B.

Disk error reading FAT (x)
[Chkdsk]

One of your File Allocation Tables has a defective sector in it. MS-DOS automatically uses the other FAT. It is a good idea to copy all your files onto another disk.

Disk error writing drive (x:)
[MS-DOS]

Device error. See Appendix B.

Disk error writing FAT (x:)
[Chkdsk]

One of your File Allocation Tables has a defective sector in it. MS-DOS will automatically use the other FAT. It is a good idea to copy all your files onto another disk.

Diskettes compare OK
[Diskcomp]

The disks that you are comparing are identical.

Disk full. Edits lost
[EDLIN]

EDLIN was not able to save your file due to lack of disk space.

Disk full--write not completed
[EDLIN]

You gave the End (E) command, but the disk did not contain enough free space for the file. EDLIN aborted the End command and returned you to the operating system. Part of the file may have been written to disk and saved. Delete the saved portion and restart the editing session. The file will not be available after this error.

Disks must be the same size
[Diskcopy]

You cannot copy the contents of a disk with a different format using Diskcopy. Use the Copy command to copy files onto the disk.

Disk unsuitable for system disk
[Format]

The Format program detected a bad track on the disk where system files should reside. You should use this disk to store data only.

Divide error
[Attrib]

The 8086 has set the divide overflow flag. This is usually caused by a program attempting to divide by zero.

Divide overflow
[MS-DOS]

The 8086 has set the divide overflow flag. This is usually caused by a program attempting to divide by zero.

Does (name) specify a file name
or directory name on the target
(F = file D = directory)?
[Xcopy]

Xcopy displays this prompt if the target directory does not exist.

Do not specify filename(s)
Command format: DISKCOMP d: d:[/1][/8]
[Diskcomp]

You specified an incorrect switch or gave a filename in addition to a drive name.

Do you see the leftmost 0? (Y/N)
[Mode]

Mode displays this message to help you align the test pattern on your screen.

Do you see the rightmost 9? (Y/N)
[Mode]

Mode displays this message to help you align the test pattern on your screen.

Drive letter must be specified
[Format]
You must specify the drive that you want to format.

Drive types or diskette types not compatible
[Diskcomp]
You cannot compare a single-sided disk with a double-sided disk, or a high-density disk with a low-density disk. You should use FC to compare the files on the disks.

Duplicate file name or File not found
[Chkdsk][MS-DOS]
You tried to rename a file to a filename that already exists or the name you specified could not be found.

ECHO is off (or on)
[MS-DOS]
This message tells you the current status of Echo.

End of input file
[EDLIN]
The entire file was read into memory. If the file is read in sections, this message indicates the last section of the file is in memory.

Enter current Volume Label for drive (x:)
[Format]
Format asks you to enter the current volume label for verification before it formats the hard disk in the specified drive.

Enter new date:
[MS-DOS]
You must respond to this prompt when you start MS-DOS. Enter the date in a mm/dd/yy format.

Enter new time:
[MS-DOS]
You must respond to this prompt when you start MS-DOS. Enter the time in the hh:mm format.

Entry error
[EDLIN]

The last command you typed contained a syntax error. Retype the command with the correct syntax and press the Return key.

Entry has a bad attribute (or link or size)
[Chkdsk]

This message may be preceded by one or two periods which indicate which subdirectory is invalid. If you have specified the /f switch, Chkdsk tries to correct the error.

Error in .EXE file
[MS-DOS]

The .EXE file you have asked MS-DOS to load has an invalid internal format.

Error reading/writing partition table
[Format]

Format could not read or write the partition table. You should run Fdisk on the disk and then try formatting it again.

Errors found, F parameter not specified
Corrections will not be written to disk
[Chkdsk]

Chkdsk found errors on the disk. If you have not specified the /f switch, Chkdsk continues printing messages but will not correct the errors.

Errors on list device indicate that it
may be off-line. Please check it.
[Print]

Your printer is not turned on.

Error writing to device
[MS-DOS]

You tried to send too much data to a device. MS-DOS was unable to write the data to the specified device.

EXEC failure
[MS-DOS]

MS-DOS either found an error when reading a command or the Files command in the CONFIG.SYS file is set too low. Increase the value and restart MS-DOS.

FCB unavailable reading drive (x:)
[MS-DOS]

Device error. See Appendix B.

FCB unavailable writing drive (x:)
[MS-DOS]

Device error. See Appendix B.

fc: cannot open (filename) - No such file or directory
[FC]

One of the files that you specified doesn't exist. Check the directory for the correct filename.

fc: (filename) longer than (filename)
[FC]

After reaching the end of one of the files in a file comparison, the other file still has uncompared data left.

fc: incompatible switches
[FC]

You have specified switches that are not compatible. (For example, /b and /L.) You should not combine binary and ASCII comparison switches.

fc: out of memory
[FC]

You do not have enough memory to perform the comparison.

fc: no differences encountered
[FC]

The files are the same.

File allocation table bad
[MS-DOS]

The disk may be defective. Run Chkdsk with /f to fix the disk.

File allocation table bad drive (x:)
[Chkdsk][Print]

The disk may be defective. Run Chkdsk with /f to fix the disk.

File cannot be copied onto itself
[Copy][Replace][Xcopy]
The source filename you specified is the same as the target filename.

File cannot be converted
[Exe2bin]
The input file is not in the correct format.

File creation error
[MS-DOS][Xcopy]
You tried to add a new filename or replace a file that already exists in the directory. If the file already exists, it is a read-only file and cannot be replaced. Run Chkdsk on the disk to determine the cause of the error.

File is READ-ONLY
[EDLIN]
You may not change this file because the file is designated read-only.

Filename must be specified
[EDLIN]
You did not specify a filename when you started EDLIN.

File not found
[EDLIN][MS-DOS][Print][Recover]
MS-DOS cannot find the file that you specified. Check to see that the pathname is accurate and that the file exists in the directory you specified.

FIND: Access denied
[Find]
You cannot access the file. Make sure that the disk is not write-protected.

FIND: File not found
[Find]
You specified a file that doesn't exist. Make sure you have typed the filename correctly.

FIND: Invalid number of parameters
[Find]

You specified too many, or not enough options in the command line.

FIND: Invalid Parameter
[Find]

One of the switches that you have specified is wrong.

FIND: Read error in (filename)
[Find]

The program could not read the specified file.

FIND: Syntax error
[Find]

Check to make sure that you have typed the command correctly.

First cluster number is invalid, entry truncated
[Chkdsk]

The file directory entry contains an invalid pointer to the data area. If you specified the /f switch, the file is truncated to a zero-length file.

FIRST diskette bad or incompatible
[Diskcomp]

Diskcomp cannot recognize the format on the source disk. You should run Chkdsk to help you identify the problem.

Fixups needed - base segment (hex:)
[Exe2bin]

The source (.EXE) file contained information indicating that a load segment is required for the file. Specify the absolute segment address where the finished module is to be located.

For cannot be nested
[MS-DOS]

Nesting of For commands is not allowed in a batch file.

Format another (Y/N)?
[Format]

Type Y (for Yes) to format another disk. Type N (for No) if you do not want to format another disk. If you accidentally type Y, you can abort the format process by typing Control-C in response to the "Strike any key to begin formatting" message.

Format complete
[Format]

Format displays this message when it has finished formatting the disk in the specified drive.

Format failure
[Format]

MS-DOS could not format the disk. This message is usually displayed with an explanation as to why MS-DOS could not format the disk.

Format not supported on drive (x:)
[Format]

You cannot use Format to format this drive.

General failure reading drive (x:)
[MS-DOS]

Device error. See Appendix B.

General failure writing drive (x:)
[MS-DOS]

Device error. See Appendix B.

Graphics characters loaded
[Graftabl]

The Graftabl command displays this message after it loads the table of graphics characters into memory.

Graphics characters already loaded
[Graftabl]

The Graftabl command displays this message if you have already loaded the table of graphics characters into memory.

Head: (hh) Cylinder: (cc)
[Format]

Format displays the head and cylinder number of the track currently being formatted.

Illegal device name
[Mode]

Your computer does not recognize this device name.

Incompatible system size
[Sys]

The system files IO.SYS and MS-DOS.SYS occupy more space on the source disk than is available on the destination disk.

Incorrect DOS version
[Attrib] [Chkdsk] [Diskcomp] [Diskcopy] [EDLIN]
[Format] [Graftabl] [Mode] [More] [Print] [Recover]
[Redir] [Share] [Subst] [Sys] [Tree] [Xcopy]

Many version 2.x and 3.x utilities will not run on earlier versions of MS-DOS. Some utilities will only run under the exact version of MS-DOS for which they were configured.

Incorrect number of parameters
[Join] [Subst]

You specified too many or too few options in the command line.

Incorrect parameter
[Assign] [Redir] [Share]

One of the options you specified is wrong.

Infinite retry on parallel printer timeout
[Mode]

Your printer is probably off-line or not ready. If the printer appears to be ready, you should press the Control-Alt-Del keys to reset the computer.

Insert destination disk in drive (x:)
and strike any key when ready
[Sys]

This message appears when you are using Sys to transfer the operating system with a single disk drive. You should insert a disk in the appropriate drive and press any character or number key to begin processing.

Insert diskette for drive (x:)
and strike any key when ready
[MS-DOS]

This message appears when MS-DOS is copying files. You should insert a disk in the appropriate drive and press any character or number key to begin processing.

Insert diskette with batch file
and press any key when ready
[MS-DOS]

The disk containing the batch file you specified is not in the drive you originally specified. Reinsert the disk that contains the batch file in the appropriate drive.

Insert DOS disk in drive (x:)
and strike ENTER when ready
[Format]

You have specified Format /s but the disk in the default drive does not contain MS-DOS system files. Insert a system disk in the drive specified, and press the Return key.

Insert FIRST diskette in drive (x:)
[Diskcomp]

Diskcomp displays this message to prompt you to insert the first disk into the specified drive.

Insert new diskette for drive (x:)
and strike ENTER when ready
[Format]

This message appears when you are using a single drive to format a disk. You should insert a disk in the appropriate drive and press the Return key to begin formatting. If there is any data on the disk, Format will destroy it.

Insert SECOND diskette in drive (x:)
[Diskcomp]

Diskcomp displays this message to prompt you to insert the second disk that you want to compare into the specified drive.

Insert source diskette into drive (x:)
[Diskcopy]

Insert the disk to be copied into the specified drive.

Insert system diskette in drive (x:)
and strike any key when ready
[Sys]

Sys needs a disk from which to read the IO.SYS and MS-DOS.SYS files. Insert a disk with the IO.SYS and MS-DOS.SYS files into the specified drive and press any character or number key to start the system copy process.

Insert target diskette into drive (x:)
[Diskcopy]

You are running Diskcopy and your source and destination drives are the same. Reinsert the destination disk into the specified drive.

Insufficient disk space
[MS-DOS] [Replace]

The disk is full. It does not contain enough room to perform the specified operation.

Insufficient memory
[Chkdsk] [Diskcomp] [EDLIN] [Replace]

There is not enough memory to perform the specified operation.

Insufficient memory for system transfer
[Format]

Your memory configuration is insufficient to transfer the MS-DOS system files IO.SYS and MS-DOS.SYS with the /s switch.

Insufficient room in root directory.
Erase files in root and repeat CHKDSK
[Chkdsk]

Chkdsk always recovers lost files into the root directory. In this case, your root directory is full. ,Delete some files in your root directory to make room for the lost files.

Intermediate file error during pipe
[MS-DOS]

The pipe operation makes use of temporary files on the disk which will automatically be deleted after the piping process is complete. An error has occurred in one of these files.

Invalid baud rate specified
[Mode]

You have specified an incorrect baud rate. Valid choices are 110, 150, 300, 600, 1200, 2400, 4800, and 9600. You must specify at least the first two digits of the baud rate.

Invalid characters in volume label
[Format][Label]

The volume label should contain only up to 11 alphanumeric characters.

Invalid COMMAND.COM
Insert COMMAND.COM disk in default drive
and strike any key when ready
[MS-DOS]

The program you have just run used up almost all of memory. MS-DOS must now reload the COMMAND.COM file from disk. However, MS-DOS cannot find COMMAND.COM on the disk or the copy found is invalid. Insert a disk into the default drive which contains a copy of COMMAND.COM similar to the version on the disk with which you started MS-DOS.

Invalid country code
[MS-DOS]

You have specified a country number in your CONFIG.SYS file which is not in the table of files configured in this implementation of MS-DOS. Country codes must be in the range 1-99 and are set by your computer manufacturer.

Invalid date
[Backup][Date][Restore][Xcopy]

You specified the date incorrectly. Refer to Chapter 3 for the correct syntax and try the command again.

Invalid device
[MS-DOS]

The device specified was not CON, NUL, AUX, or PRN.

Invalid device parameters from device driver
[Format]

Format displays this message when the number of hidden sectors is not evenly divisible by the number of sectors per track (i.e., the partition does not start on a track boundary). This might happen if you tried to format a hard disk that previously had been formatted with MS-DOS 2.x without first running Fdisk.

Invalid directory
[MS-DOS]
The directory you specified either does not exist or is invalid. Check to see that you entered the directory name correctly.

Invalid disk change reading drive (x:)
[MS-DOS]
Device error. See Appendix B.

Invalid disk change writing drive (x:)
[MS-DOS]
Device error. See Appendix B.

Invalid drive in search path
[MS-DOS]
The drive does not exist.

Invalid drive or file name
[EDLIN][Recover]
Specify a valid drive or a valid filename.

Invalid drive specification
[Chkdsk][Diskcomp][Diskcopy][Format]
[Print][Replace][Sys][Tree][Xcopy]
You must specify a valid drive.

Invalid environment size specified
[Command]
You gave an invalid number of bytes with the /e switch. You must specify a number between 128 and 32768.

Invalid number of parameters
[MS-DOS][Attrib][Recover][Xcopy]
You have specified the wrong number of options in the command line.

Invalid parameter
[Chkdsk][Diskcomp][Diskcopy][EDLIN]
[Format][Mode][Print][Replace][Sort]
[Subst][Sys][Tree][Xcopy]

One of the switches that you have specified is wrong.

Invalid path or file name
[Copy][Xcopy]

Specify a valid pathname or filename for this command command.

Invalid path, not directory, or directory not empty
[MS-DOS]

You are unable to remove the directory requested for one of the specified reasons.

Invalid sub-directory entry
[Chkdsk]

The subdirectory that you specified either does not exist or is invalid. Check to see that you typed the subdirectory name correctly.

Invalid time
[Backup][Restore][Time][Xcopy]

You specified an invalid time. Refer to Chapter 3 for the correct syntax and try the command again.

Invalid Volume ID
[Format]

Format displays this message if you enter a volume label that doesn't match the label on the hard disk you want to format. It then and quits the format process.

Invalid working directory
[MS-DOS]

Your disk is bad. Replace the disk or make another copy from your backup system disk.

Label not found
[MS-DOS]

There is a GOTO command to a nonexistent label in a batch file.

Line too long
[EDLIN]

During a Replace command, the string given as the replacement caused the line to expand beyond 253 characters. Divide the long line into two lines and retry the Replace command.

List output is not assigned to a device
[Print]

When you first run print, it asks you what device you want to specify as a print spooler. This message appears if Print is set up for a device that does not exist.

Lock violation reading drive (x:)
[MS-DOS]

Device error. See Appendix B.

Lock violation writing drive (x:)
[MS-DOS]

Device error. See Appendix B.

LPT#: not redirected
[Mode]

Mode could not redirect the parallel printer port. Check to see that you have specified the proper options.

LPT#: redirected to COM#:
[Mode]

Output on the parallel printer port will now be sent to this asynchronous communications port.

LPT#: set for 132
[Mode]

The parallel printer port has been set for 132 columns.

LPT#: set for 80
[Mode]

The parallel printer port has been set for 80 columns.

Memory allocation error. Cannot load MS-DOS, system halted
[MS-DOS]

Restart MS-DOS. If this error persists, make a new copy of the MS-DOS disk from your backup copy of the system disk.

--More--
[More]

Press the Spacebar to view more of the file or directory.

MORE: Incorrect DOS version
[More]

More will not run on versions of MS-DOS previous to 2.0.

Must specify destination line number
[More]

You must specify a destination line number when you are copying and inserting lines with EDLIN.

Must specify ON or OFF
[MS-DOS]

The command requires either an ON or an OFF argument.

Name of list device [PRN]:
[Print]

This prompt appears the first time that Print is run. Any valid device may be specified and that device then becomes the Print output device.

Network has not been started
[Redir]

You must start the network before you can use the Redir command.

New file
[EDLIN]

This message is printed if EDLIN does not find a file with the name you specified. If you are creating a new file, ignore this message. If you do not intend to create a new file, check to see that you correctly typed the filename of the file you wish to edit.

No COM: ports
[Mode]

Your computer does not have a COM: port.

No files added/replaced
[Replace]

The Replace command did not add or replace any files.

No files found (filename)
[Replace]

Replace could not find matching source or target files.

No files match d:xxxxxxxx.xxx
[Print]

A filename was given for files to add to the queue, but no files match the specification.

No free file handles.
Cannot start COMMAND.COM, exiting
[MS-DOS]

Restart MS-DOS. If this message persists, increase the Files command value in the CONFIG.SYS file.

Non-DOS disk error reading drive (x)
[MS-DOS][Print]

Device error. See Appendix B.

Non-system disk or disk error
Replace and strike any key when ready
[Format][Sys]

Replace the disk with the proper disk and press any alphanumeric key to continue.

No paper error writing device (dev)
[MS-DOS]

Device error. See Appendix B.

No path
[MS-DOS]

You typed Path and pressed the Return key to find out what your search path is. There is no current command search path.

No room for system on destination disk
[Sys]

There is not enough room for the system files on the destination disk. Delete some files to make room for the system files or use another disk. You may need to reformat the disk to put the system on it.

No room in directory for file
[EDLIN]

You have tried to save a file to the root directory but it is full. Subdirectories are not limited in size as is the root directory.

No subdirectories exist
[Tree]

The disk in the drive you specified does not contain subdirectories.

No system on default drive
[Sys]

The system files do not exist on the disk in the default drive.

Not enough memory
[Join][Share][Subst]

There is not enough memory for MS-DOS to run the command.

Not enough room to merge the entire file
[EDLIN]

There was not enough room in memory to hold the file during a Transfer command. You must free some memory by writing some files to disk or deleting some files before you can transfer this file.

Not found
[EDLIN]

You have specified a Search or a Replace command that was unable to find a further occurrence of the specified Search or Replace string.

Not ready error reading drive (x:)
[MS-DOS]

Device error. See Appendix B.

Not ready error writing drive (x:)
[MS-DOS]
Device error. See Appendix B.

O.K.?
[EDLIN]
This prompt occurs during Search and Replace command processing. If you press any key except Y or the Return key, the search or replace process continues.

Out of environment space
[MS-DOS]
There is not enough room in the program environment to accept more data. You can use Command with the /e switch to increase the size of the environment.

Parameters not compatible
[Format][Replace]
You have specified switches that cannot be used together.

Parameters not compatible with fixed disk
[Format]
You have used a switch that is not compatible with the specified drive.

Parameters not supported
[MS-DOS][Format]
You have specified parameters that MS-DOS does not support.

Parameters not supported by Drive
[Format]
Format displays this message when the device driver for this drive does not support Generic IOCTL function requests.

Path not found
[Chkdsk][Replace][Subst]
You specified an invalid path.

Path too long
[Replace] [Xcopy]
The path name you specified was too long. You may have to change directories to replace files in deep subdirectories.

Press any key to begin adding files
[Replace]
When you specify the /w switch, Replace displays this message to prompt you to start replacing files.

Press any key to begin recovery of the file(s) on drive (x:)
[Recover]
This prompt is issued before you recover a disk or file. Press any character or number key to begin the recover process. If you wish to end this command, press Control-C.

Press any key when ready
[Diskcomp] [Diskcopy]
This prompt occurs when you are copying disks. When you have inserted the disks into the appropriate drives, press any character or number key to begin the Diskcopy process. If you want to end this command, press Control-C.

Printer lines per inch set
[Mode]
Mode has set the number of lines per inch for the printer.

PRINT queue is empty
[Print]
There are no files waiting to be printed.

PRINT queue is full
[Print]
There is only room for 10 files in the list of files waiting to be printed.

Probable non-DOS disk
Continue (Y/N)?
[Chkdsk]
The disk you are using is not recognized by this version of MS-DOS. The disk either was created by another system with a format that is not supported

on this version of MS-DOS or is not an MS-DOS disk. Do not continue processing if Chkdsk returned this message for a removable disk. If this message is returned for a hard disk, the information describing the characteristics of the disk to MS-DOS has been destroyed. In this case, you may continue Chkdsk processing.

Processing cannot continue
[Chkdsk]

There is not enough memory in your machine to process Chkdsk for this disk. You must obtain more memory to run Chkdsk.

Program too big to fit in memory
[MS-DOS]

You must acquire more memory to run your application. It is possible that some programs you have run are still using some memory. You may try to restart MS-DOS; however, if you still receive this message, you must acquire more memory.

Read error in (filename)
[Find]

MS-DOS could not read the file.

Read fault error reading drive (x:)
[MS-DOS]

Device error. See Appendix B.

Reading source file(s)...
[Xcopy]

Xcopy is now reading the source files that you specified.

Redirector already installed
[Redir]

The Redirector can only be installed once.

Re-insert diskette for drive (x:)
[Format]

Reinsert the disk being formatted in the indicated drive.

Resident part of PRINT installed
[Print]

This is the first message that MS-DOS displays when you issue the Print command. It means that available memory has been reduced by several thousand bytes to process the Print command concurrent with other processes.

Resident portion of MODE loaded
[Mode]

Part of the Mode program is now resident in memory.

Resynch failed. Files are too different.
[FC]

FC displays this message if the number of lines in the internal line buffer is less than the number of consecutive, differing lines. You should specify the /Lb switch with a larger number if you want to display all of the file differences.

SECOND diskette bad or incompatible
[Diskcomp]

The second disk does not contain the same format as the first disk, or Diskcomp does not recognize the format of the second disk. You should run Chkdsk to help you identify the problem.

Sector not found error reading drive (x:)
[MS-DOS]

Device error. See Appendix B.

Sector not found error writing drive (x:)
[MS-DOS]

Device error. See Appendix B.

Sector size too large in file (filename)
[MS-DOS]

The specified device driver loaded by CONFIG.SYS uses a sector size larger than that of any other device driver on the system. You cannot run this device driver.

Seek error reading drive (x:)
[MS-DOS]

Device error. See Appendix B.

Seek error writing drive (x:)
[MS-DOS]
Device error. See Appendix B.

SHARE already installed
[Share]
Share can only be installed once.

Sharing violation reading drive (x:)
[MS-DOS]
Device error. See Appendix B.

SORT: Incorrect DOS version
[Sort]
Sort will not run on MS-DOS versions prior to MS-DOS 2.0.

SORT: Insufficient disk space
[Sort]
The disk is full.

SORT: Insufficient memory
[Sort]
There is not enough memory to run the Sort program.

Source and target diskettes are not the same format.
Cannot do the copy
[Diskcopy]
You must have the same size and kind of disks to run Diskcopy. Example: you cannot copy from a single-sided disk to a double-sided disk. Reformat the target disk to be of the same type as the source disk.

Source path required
[Replace]
You did not specify a source path for Replace.

Specified drive does not exist,
or is non-removable
[Diskcomp]
You must specify the name of a valid floppy drive. Diskcomp cannot compare hard disks.

Specified MS-DOS search directory bad
[MS-DOS]

The Shell command in the CONFIG.SYS file is incorrect. The place that you have told MS-DOS to find COMMAND.COM does not exist, or COMMAND.COM is not in that place.

Strike a key when ready...
[MS-DOS]

This prompt occurs during command processing and is always accompanied by another message. This message is also displayed if you have inserted a Pause command in a batch file. Usually, you are asked to insert disks into appropriate drives before this prompt. Press any character or number key to begin command processing.

Syntax error
[MS-DOS][Find]

Check to make sure that you have typed the command correctly.

System transferred
[Format][Sys]

The system files MS-DOS.SYS and IO.SYS have been transferred during Format or Sys command processing.

Terminate batch job (Y/N)?
[MS-DOS]

If you press Control-C while in batch mode, MS-DOS asks you whether or not you wish to end batch processing. Press Y to end processing. Press N to continue the batch job.

Too many files open
[EDLIN]

MS-DOS could not open the file to edit on the .BAK file due to lack of system file handles. Increase the value of the Files command in the CONFIG.SYS file.

Track 0 bad - disk unusable
[Format]

The Format command can accommodate for defective sectors on the disk except ror those near the beginning. In this case, use another disk.

Unable to create a directory
[MS-DOS][Xcopy]

MS-DOS could not create the directory you specified. Check to see that there is not a name conflict (you may have a file by the same name) or the disk may be full.

Unable to shift Screen
[Mode]

Mode is unable to shift the test pattern on the screen any farther.

Unexpected DOS Error (NNN)
[Replace]

An unexpected error occurred, where (NNN) is the MS-DOS error number.

Unrecognized command in CONFIG.SYS
[MS-DOS]

There is an invalid command in your CONFIG.SYS file. Refer to Appendix D for a list of valid statements.

Unrecoverable error in directory
Convert directory to file (Y/N)?
[Chkdsk]

If you respond Y to this prompt, Chkdsk will convert the bad directory into a file. You can then fix the directory yourself or delete it.

usage: fc [/a] [/b] [/c] [/l] [/lb n] [/w]
[t] [/n] [/NNNN] file1 file2
[FC]

One of the switches that you have specified is invalid.

VERIFY is off (or on)
[MS-DOS]

This message tells you the current setting of the Verify command.

Volume in drive (x:) has no label
[MS-DOS][Label]

This is an informational message displayed in response to the Dir and Label commands.

Volume in drive (x:) is (label)
[MS-DOS][Label]
This is an informational message displayed in response to the Dir and Label commands.

Volume label (11 characters, ENTER for none)?
[Format][Label]
This message is displayed when you specify the /v switch in the Format command, or when you give the Label command. Specify a volume label or press the Return key to indicate that you do not want a volume label for this disk.

WARNING, ALL DATA ON NON-REMOVABLE DISK
DRIVE X WILL BE LOST!
Proceed with Format (Y/N)?
[Format]
There is data on the hard disk that you are trying to format. If you want to lose the data and format the disk, press Y (for Yes). If you do not want the files on your hard disk erased, press N (for No). Copy the files to a floppy disk and repeat the Format command.

Warning - directory full
[Recover]
The root directory is too full for Recover processing. Delete some files in the root directory to free space.

Warning: Read error in EXE file
[Exe2bin]
The amount read was less than the size of the header. This is a warning message only.

Write fault error writing drive (x:)
[MS-DOS]
Device error. See Appendix B.

Write protect error writing drive (x:)
[MS-DOS]
Device error. See Appendix B.

Appendix F
Configuring Your Hard Disk (FDISK)

APPENDIX F

CONFIGURING YOUR HARD DISK (FDISK)

F.1 INTRODUCTION

Hard disks can be divided into one to four separate sections, called partitions. Partitions separate your hard disk into individual areas, and each partition may contain a different operating system.

To prepare your hard disk for MS-DOS, you must create a partition for MS-DOS, called a DOS partition. You can create a DOS partition on your hard disk by using a menu-driven utility called Fdisk. You need to use Fdisk if you want to:

- use your hard disk for MS-DOS
- create a DOS partition
- change the active partition
- delete a DOS partition
- display partition information
- choose the next hard disk on your computer

If you want to use your entire hard disk for MS-DOS, you will only use the Fdisk program once to create the DOS partition.

Note

Many computer stores configure hard disk computers for MS-DOS, so you may not need to use Fdisk. They may also format your hard disk to start MS-DOS when you turn the power on. To determine whether or not this has been done, try to start MS-DOS from your hard disk, or run Fdisk and try to create a DOS partition. Fdisk displays a message if your hard disk already has a DOS partition.

F.2 STARTING FDISK

To start Fdisk, type:

```
fdisk
```

and press the Return key. Fdisk displays its main menu on your screen:

```
Fixed Disk Setup Program Version 0.01
(C)Copyright Microsoft, 1985, 1986.

FDISK Options


Choose one of the following:

    1.  Create DOS Partition
    2.  Change Active Partition
    3.  Delete DOS Partition
    4.  Display Partition Data
    5.  Select Next Fixed Disk Drive

Enter choice:[1]


Press ESC to return to DOS
```

If your computer does not have more than one hard disk, choice 5 will not appear on this menu.

The following sections contain a description of each of these options, and show the screens that they display. To return to MS-DOS from the main menu, just press the Escape key. You can also use the

Escape key to return to the main menu from any of the other Fdisk screens.

Each Fdisk screen displays a default value. To choose the default value, press Return. To choose another value, just type the value you want, and then press the Return key.

F.2.1 Choice 1 - Creating a DOS Partition

If you choose the first option on the main menu, and if your hard disk already contains a DOS partition, Fdisk displays a screen showing partition status information for your hard disk. For example, your screen might look something like this:

```
Create DOS Partition


Partition Status  Type   Start  End Size
    1       A      DOS       0  304  305

Total disk space is 305 cylinders

Fixed disk already has a DOS partition.



Press ESC to return to FDISK Options
```

If your hard disk does not contain a DOS partition, Fdisk displays the following screen:

```
Create DOS partition

Current Fixed Disk Drive: 1


Do you wish to use the entire fixed disk
for DOS (Y/N)....................? [Y]
```

If you want to use the entire hard disk for MS-DOS, press the Return key to accept the default setting, Y. Fdisk displays the message:

```
System will now restart
Insert DOS diskette in drive A:
Press any key when ready . . .
```

You should insert your MS-DOS disk in drive A and press any key to restart MS-DOS.

Now that you have created your DOS partition, you must format your hard disk so that MS-DOS can use it. If you want to start MS-DOS from your hard disk, remember to use the Format /s switch. (Refer to the Format command in Chapter 3 "MS-DOS Commands," for more information on how to format your hard disk.)

F.2.1.1 Using Part of Your Hard Disk for MS-DOS -

If you want to use only part of your hard disk for MS-DOS, press N, then press Return. Fdisk displays the following message:

```
Total fixed disk space is xxxx cylinders
Maximum available space is xxxx.
cylinders at xxxx.
Enter partition size............: [xxxx]
```

This message shows the total number of cylinders available for a hard disk partition, and prompts you to enter the size of your new partition. The default for the partition size is the maximum available space on the hard disk. Press the Return key if you want the default; otherwise type the size (in cylinders) that you want for the partition, and press the Return key. Fdisk now displays:

```
Enter starting cylinder number..:[xxxx]
```

Note

Fdisk adjust the partition boundaries to avoid bad tracks if it finds any defective tracks at the start of the partition.

The default starting cylinder number starts at the first cylinder large enough to contain the partition. If you want to locate the DOS partition at the default location, press the Return key. Otherwise, type the number of the cylinder where you want the DOS partition, and press the Return key.

If you want to start MS-DOS from this partition on your hard disk, remember to use the Format /s switch. (Refer to the Format command in Chapter 3 "MS-DOS Commands," for more information on formatting your hard disk.)

Note

You can only have one DOS partition on your hard disk at a time. To create a partition for another operating system you must refer to the documentation for that operating system.

F.2.2 Choice 2 - Changing the Active Partition

If you choose the second option on the main menu, Fdisk displays a screen showing the status of each partition on your hard disk. The active partition, indicated with a status of "A", is the partition whose operating system and files you access when you turn on the power or reset your computer. Only one partition is active at a time, the others are non-active, indicated with a status of "N".

For example, your Change Active Partition screen might look like this:

```
Change Active Partition

Current Fixed Disk Drive: 1

Partition Status  Type     Start End  Size
    1        A     DOS        0  119  120
    2        N   non-DOS    120  304  185


Total disk space is   305  cylinders

Enter the number of the partition you
want to make active...............: [1]


Press ESC to return to FDISK Options
```

Type the number of the partition that you want to activate, and press the Return key. The default setting is the active partition number.

If you are using the entire hard disk as a DOS partition, Fdisk displays the message:

```
Partition 1 is already active

Press Esc to return to FDISK Options
```

instead of prompting you for the partition that you want to activate. Press the Esc key to return to the main menu.

F.2.3 Choice 3 - Deleting a DOS Partition

If you choose the third option on the main menu, Fdisk displays a screen showing the status of each partition on your hard disk, and prompts you to delete the DOS partition. When you delete a DOS partition, Fdisk deletes the partition boundaries and any data that existed in that partition. Once you delete the partition, YOU CANNOT RECOVER THAT DATA.

Note

You cannot use Fdisk to delete a non-DOS partition. You need to put an MS-DOS system disk into drive A to continue using MS-DOS after you have deleted the DOS partition. If you want to start a different operating system in another partition of your hard disk, you need to change the active partition to that number BEFORE you delete the DOS partition.

Your Delete DOS Partition screen might look like this:

```
Delete DOS Partition

Current Fixed Disk Drive: 1


Partition Status  Type   Start End Size
    1        N     DOS      0  304  305

Total fixed disk space is 305 cylinders

Warning! Data in the DOS partition
will be lost.  Do you wish to
continue.......................? [N]

Press ESC to return to FDISK Options
```

If you do not want to delete the DOS partition, press the Return key to accept the default value N. To delete the DOS partition, type Y, and then press the Return key.

F.2.4 Choice 4 - Displaying Partition Data

If you choose the fourth option on the main menu, Fdisk displays a screen that contains information about each of the partitions on your hard disk.

For example, your Display Partition Information screen might look like this:

```
Display Partition Information


Partition Status   Type    Start End Size
    1        N      DOS        0  304  305


Total fixed disk space is 305 cylinders


Press ESC to return to FDISK Options
```

This screen gives you information about the number of the partition, the status of the partition, the type of the partition, the starting and ending cylinder numbers of the partition, and the size of the partition in cylinders. Press the Esc key to return to the main menu.

F.2.5 Choice 5 - Next Fixed Disk

This option appears on the Fdisk main menu only if you have more than one hard disk attached to your computer. If you choose this option, Fdisk changes the current disk drive to the next drive number.

For example, if the current disk drive is number 1, and you choose option 5 on the main menu, Fdisk changes the current disk drive to number 2. Now you can choose any of the Fdisk options to prepare the second fixed disk for MS-DOS.

Appendix G
Installable Device Drivers

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

APPENDIX G

INSTALLABLE DEVICE DRIVERS

G.1 INTRODUCTION

DRIVER.SYS and RAMDRIVE.SYS are two installable device drivers provided with MS-DOS. This appendix explains the details of these drivers. You should read Appendix D, "How to Configure Your System" for more information on how to install them.

G.2 DRIVER.SYS

DRIVER.SYS is an installable device driver that supports external drives. To install DRIVER.SYS, include the following command line in your CONFIG.SYS file:

```
device=driver.sys /d:<dd> [/c] [/f:<ff>] [/h:<hh>] [/n]
[/s:<sss>] [/t<ttt>]
```

The <dd> parameter on the /d switch specifies a physical drive number between 0 and 255. Physical drives are numbered differently than logical drives. Floppy drives are numbered starting at 0, and hard drives are numbered starting at 80H.

For example, if you have a computer with two floppy drives, they might be numbered 0 and 1. If you add an external floppy drive, its physical drive number is 02H.

If you have a computer with one floppy and one hard drive, the floppy drive is number 00H, but the hard drive is number 80H. If you add an external floppy drive, its physical drive number is still 02H, because MS-DOS already knows the definitions of physical drives 00H and 01H.

The /c switch specifies that changeline (doorlock) support is required.

The <ff> option on the /f switch specifies the form factor index where:

```
0 = 320/360 KB
1 = 1.2 MB
2 = 720 KB
3 = 8" single density
4 = 8" double density
5 = Hard disk
6 = Tape drive
7 = Other
```

If you do not specify the /f switch, DRIVER.SYS uses a default of 720 KB.

The <hh> option on the /h switch specifies the maximum head number. Its value can range from 1 to 99.

The /n switch specifies a non-removable block device.

The <ss> option on the /s switch specifies the number of sectors per track. Its value can range from 1 to 99.

The <ttt> option on the /t switch specifies the number of tracks per side on the block device. Its value can range from 1 to 999.

For example, if you want to add an external 720 KB drive to your computer, you would include the following line in the CONFIG.SYS file:

```
device=driver.sys /d:02
```

G.3 RAMDRIVE.SYS

RAMDRIVE.SYS is a device driver that lets you use a portion of your computer's memory as if it were a disk drive. This memory area is called a RAM disk and is sometimes referred to as a virtual disk.

RAM disks are much faster than disk drives because the information they contain is always loaded into memory. If your computer has extended memory installed (starting at the 1MB boundary), or if you have an extended memory board that meets the Lotus(R)/Intel(R)/Microsoft(R) Expanded Memory Specification, you can use this memory for one or more RAM disks. Otherwise RAMDRIVE.SYS locates RAM drives in low memory.

Note

This command increases the size of MS-DOS resident in memory.

To install RAMDRIVE.SYS, include the following command in your CONFIG.SYS file:

```
device=ramdrive.sys [<bbbb>] [<ssss>] [<dddd>] [/e | /a]
```

The <bbbb> option specifies the disk size in kilobytes. The default is 64, the minimum value is 16.

The <ssss> option specifies the sector size in bytes. The default value is 128. The values: 128, 256, 512, and 1024 are allowed.

The <dddd> option specifies the number of root directory entries. The default value is 64, the minimum value is 2, and the maximum value is 1024.

RAMDRIVE.SYS adjusts the value of <dddd> to the nearest sector boundary. For example, if you give a value of 25 when the sector size is 512 bytes, the 25 will be rounded up to 32. This is because 32 is the next multiple of 16 (there are sixteen 32-byte directory entries in 512 bytes).

The /e option lets you use extended memory (above 1MB) as a RAM disk if it has been installed. If you use this switch, you cannot use the /a switch.

The /a option lets you use an extended memory board that meets the Lotus(R)/Intel(R)/Microsoft(R) Expanded Memory Specification for a RAM drive, if that board has been installed. If you use this switch, you cannot use the /e switch.

Note

When you reset or turn off the power on your computer, the information stored in RAM disks is lost.

INDEX

MICROSOFT™